ARMY RESEARCH LABORATORY

# Meteorological Sensor Array (MSA) – Phase I Volume 1 (“Proof of Concept” Overview)

**by Gail Vaucher, Jeffrey Swanson, John Raby, Theresa Foley, Sandra Harrison, Robert Brice, Sean D’Arcy, and Edward Creegan**

**ARL-TR-7058** **September 2014**

**Approved for public release; distribution is unlimited.**

**NOTICES**

**Disclaimers**

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed.  Do not return it to the originator.

**Army Research Laboratory**
White Sands Missile Range, NM 88002-5501

**ARL-TR-7058** **September 2014**

# Meteorological Sensor Array (MSA) – Phase I Volume 1 (“Proof of Concept” Overview)

**by Gail Vaucher, Jeffrey Swanson, John Raby, Theresa Foley, Sandra Harrison, Robert Brice, Sean D’Arcy, and Edward Creegan**
**Computational and Information Sciences Directorate, ARL**

**Approved for public release; distribution is unlimited.**

| REPORT DOCUMENTATION PAGE | *Form Approved* *OMB No. 0704-0188* |
|---|---|

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
**PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

| | | |
|---|---|---|
| **1. REPORT DATE** *(DD-MM-YYYY)* September 2014 | **2. REPORT TYPE** Final | **3. DATES COVERED (From - To)** October 1, 2013–September 30, 2014 |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** Meteorological Sensor Array (MSA) – Phase I, Volume 1 (“Proof of Concept” Overview) | **5a. CONTRACT NUMBER** |
| | **5b. GRANT NUMBER** |
| | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Gail Vaucher, Jeffrey Swanson, John Raby, Theresa Foley, Sandra Harrison, Robert Brice, Sean D’Arcy, and Edward Creegan | **5d. PROJECT NUMBER** |
| | **5e. TASK NUMBER** |
| | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** U.S. Army Research Laboratory, Computational and Information Sciences Directorate, Battlefield Environment Division (ATTN: RDRL-CIE-D), White Sands Missile Range, NM 88002-5501 | **8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-7058 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | **10. SPONSOR/MONITOR’S ACRONYM(S)** |
| | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
The US Army Research Laboratory (ARL) is creating a Meteorological Sensor Array (MSA) to improve Army decisions by strengthening environmentally-dependent decision aids through validated high-resolution atmospheric models in the boundary layer to provide reliable and persistent data resources, which allow modelers and sensor developers to validate model and sensor performance with atmospheric observations. This report outlines the multiphase MSA Development Plan, and documents the MSA-Phase I (*Proof of Concept*). Phase I included a) 5 equally-spaced meteorological towers located around a large Solar Photovoltaic Farm in southern NM; b) measurements of pressure, temperature (2 m/10 m), relative humidity (2 m), insolation (2 m) and winds (2 m/10 m); c) solar-powered instrumentation; and d) wireless data download, monitoring, and time synchronization. The MSA data processing included data merging, plotting and formatting for applications. Automating data quality assurance was investigated. Model validation and verification (V&V) was one of several MSA-Phase I data applications. The Weather Running Estimate-Nowcast (WRE-N) model was used to confirm the feasibility of Model V&V with MSA data. Based on the success of Phase I, the authors look forward to future, fruitful phases.

**15. SUBJECT TERMS**
meteorological sensor array, MSA, model validation and verification, atmospheric sensors, data assurance, WRE-N, 3DWF

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Gail Vaucher |
|---|---|---|---|---|---|
| **a. REPORT** Unclassified | **b. ABSTRACT** Unclassified | **c. THIS PAGE** Unclassified | UU | 114 | **19b. TELEPHONE NUMBER** *(Include area code)* (575) 678-3237 |

**Standard Form 298 (Rev. 8/98)**
**Prescribed by ANSI Std. Z39.18**

# Contents

**List of Figures** **vi**

**List of Tables** **vii**

**Acknowledgments** **ix**

**Executive Summary** **xi**

**1. Background** **1**

1.1 The Challenge.....1

1.2 Meteorological Sensor Array (MSA) Overview.....3

1.2.1 MSA Vision.....3

1.2.2 MSA Program Phases.....4

**2. Sensor Calibration** **5**

2.1 Dynamic Sensor Calibration.....5

2.2 Thermodynamic Sensor Calibration.....6

**3. Tower Design Description** **8**

**4. Tower Data Acquisition Systems (DAS)** **11**

4.1 Thermodynamic (Logger) DAS.....12

4.2 Dynamic (Sonic) DAS.....13

**5. Tower Power Design** **13**

**6. Communications** **14**

**7. Data Processing** **15**

**8. Automated Quality Assurance Protocol** **16**

8.1 QA Limits.....17

8.1.1 Battery Voltage.....17

8.1.2 Panel Temperature.....17

8.1.3 Internal Quality Control.....18

8.1.4 Climatological Average Values ........ 18
8.1.5 Air Temperature (2- and 10-m AGL) ........ 18
8.1.6 Relative Humidity (2-m AGL) ........ 18
8.1.7 Air Pressure (Surface) ........ 19
8.1.8 Incoming Solar Radiation (2-m AGL) ........ 19

8.2 Redundant Data ........ 21
8.2.1 Air Temperature (2- and 10-m AGL) ........ 22
8.2.2 Relative Humidity (2-m AGL) ........ 22
8.2.3 Air Pressure (Surface) ........ 22
8.2.4 Solar Radiation ........ 22

8.3 Spatial Consistency of Data ........ 22

**9. Data Applications 23**

9.1 V&V Tools and Preliminary Results ........ 24
9.1.1 NCAR MET Point-Stat ........ 24
9.1.2 GIS ........ 25

**10. Summary and Final Comments 25**

**11. References 28**

**Appendix A. Meteorological Sensor Array (MSA)-Phase I, CR23X Micrologger Program 29**

**Appendix B. Meteorological Sensor Array (MSA)-Phase I, Pre-Data-Merge Logger Program 35**

**Appendix C. Meteorological Sensor Array (MSA)-Phase I, Pre-Data-Merge Sonic Program 43**

**Appendix D. Meteorological Sensor Array (MSA)-Phase I, Pre-Data-Merge Averaging Program 51**

**Appendix E. Data Management Program – Meteorological Sensor Array (MSA)_ Mountain Standard Time (MST)_Automate.vbs 63**

**Appendix F. Data Management Program – MSA_QC_MST.gle 71**

**Appendix G. Data Management Program – Met_ASCII_UTC.vbs 83**

**Appendix H. Data Management Program – MET_ASCII_UTC.wbt** **93**

**List of Symbols, Abbreviations, and Acronyms** **95**

**Distribution List** **97**

## List of Figures

Fig. 1 MSA locations—Phase I field campaign was completed in 2014, the locations for Phases II and III are projected locations ....4

Fig. 2 The RM Young 81000 Ultrasonic Anemometer was the sensor quantifying the dynamic atmospheric attributes ....5

Fig. 3 Dynamic sensor side-by-side intercomparisons ....6

Fig. 4 Thermodynamic sensor side-by-side intercomparisons ....7

Fig. 5 a) MSA tower #4 at White Sands Missile Range (WSMR); NM, b) 10-m tower (Climatronics Corp 2014). Fig. 5b used with permission from David W Gilmore Met One Instruments, Inc. – Climatronics Division. ....9

Fig. 6 MSA tower environmental enclosure ....10

Fig. 7 MSA tower solar panel, batteries (covered), and power inverter ....11

Fig. 8 The integrated thermodynamic and dynamic DAS ....12

Fig. 9 Summary of the MSA-Phase I power connections ....14

Fig. 10 MSA-Phase I communications/network connections ....15

## List of Tables

Table 1 Pre-Phase I field-campaign sonic calibration position assignments. Position number 1 was the southern-most position—8 northern-most. Each number listed under a Group heading represents a specific sonic. ....6

Table 2 Sensors used in the MSA Phase I "Proof of Concept" design....8

Table 3 Climatological temperatures in degrees Centigrade, at WSMR, NM. Adapted from Weather Channel at http://www.weather.com/weather/wxclimatology/monthly/88002....18

Table 4 The 1948 to 1971 average relative humidity at WSMR from Fugate and Chambers (1972)....19

Table 5 The 1948 to 1971 average surface pressure in millibars, at the WSMR Station A, near the Headquarters Building, from Fugate and Chambers (1972) ....19

Table 6 Average solar radiation in W/m2 for each hour and month. These values are averages of 2011–2013 measurements made at the White Sands Museum. Data were obtained from the University of Utah, Department of Atmospheric Sciences Web site (University of Utah 2014). http://mesowest.utah.edu/cgi-bin/droman/mesomap.cgi?state=NM&rawsflag=3....20

Table 7 Standard deviation of solar radiation in W/m2 for each hour and month. These 2011-2013 measurements made at the White Sands Museum. Data were obtained from the University of Utah, Department of Atmospheric Sciences Web site. http://mesowest.utah.edu/cgi-bin/droman/mesomap.cgi?state=NM&rawsflag=3....21

INTENTIONALLY LEFT BLANK.

## Acknowledgments

The authors wish to thank Mr. Charlie Perez (Survivability/Lethality Analysis Directorate) for his invaluable support in the construction and maintenance of the Meteorological Sensor Array (MSA) Phase I field campaign. Appreciation is also expressed to the White Sands Missile Range, Range Operations-Telemetry Branch, Meteorological Section (Jim Wilkes, Mark Chaffin, Joshua Struiksma, and Ubaldo Soto) and High-Energy Laser Systems Test Facility (Tim Chavez) for their field campaign contributions. For his interest and support in the areas of Model Validation and Verification and Data Management topic, we thank Dr. Jeffrey Smith. To Felipe Chavez, we are most grateful for your ever efficient and knowledgeable skills in getting required administrative paperwork prepared and processed in a timely manner.

Many thanks go to the Technical Publishing Branch at White Sands Missile Range (WSMR), NM, for their consistent high standard of editing, specifically Sherry Larson.

INTENTIONALLY LEFT BLANK.

## Executive Summary

Army decisions can be improved only when the information on which the decisions are based is improved. Environmentally-dependent decisions involving the atmosphere rely heavily on accurate atmospheric models. The “Army-scale” atmospheric models include high-resolution (≤1-km) models. Locating meteorological observations to validate these high-resolution atmospheric models is very difficult. The National Research Council (NRC) recognized this critical technological gap, after they reviewed the US Weather Research, and Researcher-to-Operations progress and priorities in 2009. Numerous NRC conclusions and recommendations produced by this forum centered on this void in the atmospheric research community.

The US Army Research Laboratory (ARL) has responded to the National and Military need by proposing an observational data resource specifically designed to address the “Army-scale”, high-resolution atmospheric model validation and verification issues. The manifested solution is called the Meteorological Sensor Array (MSA).

The MSA vision is built on improving Army decisions by strengthening environmentally-dependent decision aids through validated high-resolution atmospheric models in the boundary layer. The MSA is intended to provide reliable and persistent data resources, which, in turn, allow atmospheric modelers and sensor developers to validate and compare model and sensor performance with atmospheric observations at and near the surface and in close proximity to terrain of varying complexity.

The multiphase program was initiated with a “Proof of Concept”, which included 5 equally spaced meteorological towers around a large Solar Photovoltaic (PV) Farm. Phase II will integrate Phase I (“Proof of Concept”) lessons learned, as it expands the array into a 36-tower gridded design. The project location of Phase II is projected to be a mid-valley, desert location. Phase III will mirror the 36-tower configuration, in a climatologically upwind location with a tall mountain range in between the two 36-tower arrays. Additional volume measurements will supplement the design. Phase IV will focus on mobilizing the array with supplemental sampling technology. Phase V is envisioned as having the capability of participating in remote test site field campaigns.

The MSA Phase I field campaign began with a calibration of all sensors slated for field usage. A brief description of the side-by-side sensor intercomparison configuration and results is in Section 2.

The MSA tower design included a portable lightweight aluminum tower with sensors mounted on the 2- and 10-m levels. The data acquisition systems (DAS) were divided into 2 categories: Thermodynamic and Dynamic DAS. A Campbell Scientific CR23X micrologger assimilated the Thermodynamic 1-min averaged data. The variables consisted of pressure, temperature (2- and

10-m above ground level [AGL]), relative humidity (2-m AGL), and insolation (2-m AGL). The Dynamic data originated from 2 RM Young 81000 Ultrasonic anemomemters (2- and 10-m AGL) sampling at 20 Hz. The variables acquired were wind speed, wind direction, u-component, v-component, w-component, speed of sound and sonic-temperature. The raw dynamic data were preserved in files on a laptop computer then reduced to 1-min averages. The dynamic and thermodynamic 1-min averages were merged with the thermodynamic data. A Digi Edgeport RS232 multiport adapter bridged the 2 data resources. A system clock on each tower was synchronized using the Network Time Protocol (NTP).

Electrical power for the remotely located towers came from a PV panel, charge controller and battery storage unit, specifically tailored to support the temporary Phase I tower configuration. The direct current (DC) electronics received a direct feed from the battery, whereas, the alternating current (AC) requirements of the computer and wireless adapters (communications) utilized an inverter. Future Phases will have reduced electrical tower requirements, and therefore, a modified power resource design.

The arrival of the Phase I wireless technology was delayed, due to bureaucratic approval processes. Consequently, the initial data flow required a daily “sneaker net” download of data from each tower, manual maintenance and monitoring of each tower, and a constant time synchronization check. Once the wireless technology could be installed, these 3 functions were done through a remote access.

Data processing began with the averaging and merging of thermodynamic and dynamic data. Plots of the time series of each sensor were created and reviewed, daily.

Research into automating and improving data quality assurance were investigated. One suggested improvement was to program quality assurance limits that would automatically flag potential data issues. Tables framing these limits are given in Section 8. Automating the intercomparison of neighboring sensors was also suggested.

The MSA design was intended for multiple applications. Model validation and verification (V&V) was one of the primary Phase I applications. As part of the MSA task, the method for executing a high-resolution model V&V was explored. Mesogamma and microscale models that were considered were the Weather Running Estimate-Nowcast (WRE-N) model and the Three-Dimensional Wind Field (3DWF) model. The tower locations and spacing were based on the 3DWF model; however, the model was not available during the field campaign, so that V&V implementation was postponed. Instead, the MSA observations were weighed against a WRE-N model output. Format changes to accommodate both the WRE-N model attributes (spatial and temporal scales) and the Model Evaluation Tools (MET) statistical tools were developed. Reasonable numerical results were produced, confirming the feasibility of a model grid to observation point comparison. Elaboration and interpretation of the V&V results were left for a separate publication.

Two core goals for the MSA Phase I task were to create a MSA development plan and to manifest a “Proof of Concept” from which future MSA Phases could be built. This report outlines the MSA Development Plan and documents the manifested MSA Phase I “Proof of Concept”. Based on the success of Phase I, the authors look forward to future, fruitful phases.

INTENTIONALLY LEFT BLANK.

# 1. Background

Improving Army decisions requires one to improve the information on which the decisions are based. For decisions involving the atmosphere, decision aids have been developed by the US Army Research Laboratory (ARL). These decision aids are based on atmospheric models, which are, in turn, based on algorithms. Physical science gleans these algorithms from measurements and observations. With high-quality measurements, informed decisions can ultimately be made.

One of the steps in the above cycle is to improve the atmospheric models that serve the decision aids. For the Army, this step requires observational ground truth data at various scales. Large resolution datasets for this task are available. However, for urban environments and foot Soldier tasks, the scale can quickly shift to very high-resolution (<1 km) datasets. The topic of this report centers on the search for a high-resolution atmospheric observational dataset.

## 1.1 The Challenge

One of the toughest challenges for validating high-resolution atmospheric models is finding a high-resolution (1 km or less), gridded-observation dataset that matches the model output grid. The National Oceanic and Atmospheric Administration (NOAA)/National Centers for Environmental Prediction (NCEP) Real-Time Mesoscale Analysis (RTMA) provide dataset products over the Continental United States at a horizontal grid spacing of 2.5 km. For verifying an “Army-scale” of 1 km or less horizontal model grid spacing, use of this product requires a remapping of the 1-km model output to a 2.5-km grid spacing, to achieve a common grid with the RTMA product. This process can result in a smoothing of wanted details in the high-resolution forecast-observation comparison.

The Variational Local Analysis and Prediction System (LAPS) developed by NOAA, Earth System Research Laboratory is another high-resolution gridded observation data resource. This system is a fully integrated, data assimilation and analysis system designed to integrate all types of meteorological observations using an analysis scheme to harmonize high-resolution temporal and spatial data onto a regularly spaced grid (Bennett et al. 2000). Through joint ARL and NOAA efforts, this Variational LAPS is being modified to generate an army-scale (1 km) gridded-observation output. As with the RTMA product, however, there is a potential for losing critical information during the harmonization of temporal and spatial data into a high-resolution data analysis.

The inadequacy of mesoscale observations was recognized during the 2009 National Research Council (NRC), Board on Atmospheric Sciences and Climate (BASC) Summer Study workshop on “Progress and Priorities of US Weather Research and Research-to-Operations Activities”. In this workshop, the NRC BASC identified priorities for addressing the national inadequacy as a challenge for developing accurate high-resolution mesoscale forecast models. These challenges were identified as knowledge gaps to be addressed over the next decade. The following list captures some of the many NRC conclusions and recommendations (National Research Council 2010):

- Observations remain inadequate to optimally run and evaluate most high-resolution models and determine forecast skills at various temporal and spatial scales.
- Assessing predictive skill is difficult, because deficiencies arise from the data and data assimilation process; errors can be found in the numerical representation; there can be intrinsic predictability limitations, and forecast verification methodology challenges. The board stated that there is a pressing need for Research and Development (R&D) leading to improved mesoscale data assimilation techniques in operational forecast systems. Further, the board stated that the basis for current knowledge of assimilation techniques is weakened by the inadequate mesoscale surface observations and lack of systematic measurements of the lower troposphere profiles of water vapor, temperature, and winds.
- The board concluded that improved analyses from mesoscale models using data assimilation techniques requires better knowledge of systematic errors in observations because of the inadequacies of the current mesoscale observations.
- The board stated that it is important that mesoscale observations are a focus of test beds designed to develop and introduce new ideas and procedures in environmental observation. They suggested the use of networks that combine observations from satellites, airborne sensors, and surface platforms. Another focus was to examine the role of mesoscale observations for new paradigms in the end-to-end forecast process important with respect to merging methods in Nowcasting with those of dynamic prediction 0–6 h range.
- The board stated the following high-priority mesoscale observing needs based on their assessment that there are essentially no current systematic national capabilities:
    - Planetary boundary layer height;
    - High-resolution vertical profiles of humidity and temperature
    - Improvements in the following measurements for direct and diffuse solar radiation, wind profiles, temperature profiles, surface turbulence, and near surface icing
- The board stated that one of the principal goals of the urban test bed is the meteorological and air quality measurement network (urban mesonet), which provide observations at high spatial and temporal resolutions from the urban core to the surrounding hinterland. The commonly accepted approach is to oversample in the test bed and use data denial modeling techniques to identify an optimal network design (or multiple, optimum designs). Data from the urban mesonet then provide the basis for a number of important activities in the urban test bed: development and testing of data assimilation and prediction models; model-

verification metrics; and applications where the observations themselves support various applications.

- The board stated that data assimilation, as part of the forecast system, is also important for acquiring and maintaining observing systems that provide the optimal cost-benefit ratio to different user groups and their applications. Data denial experiment can selectively withhold data from one (or more) system(s) and assess the degradation in forecast skill. Data assimilation can be used to determine the optimal mix of current and future in situ and remotely sensed measurements and for adaptive or targeted observations. It is also beneficial to understand the impacts of observing systems on model performance and the resulting forecast accuracy.
- The board stated that observational data with high temporal and spatial resolution are crucial to the understanding of atmospheric processes, providing data for assimilation in models, and evaluating and improving those models. This requires the synergistic combination of data from diverse sources. Rawinsonde, radar, satellite, and aircraft data as well as data from other sources all play complementary roles in weather research and forecasting.

Contemporary mesonets provide atmospheric measurements and can be found around the world. However, the stations are often far apart and on irregular grids. Various atmospheric field studies have been conducted that include high-resolution data sampling, but they are generally of a time-limited duration. The question remains, “Is there a long-term, high-resolution observational data resource?” This report describes ARL’s response toward the lack of high-resolution observational data for atmospheric model development, improvements, and calibration.

## 1.2 Meteorological Sensor Array (MSA) Overview

ARL is constructing a MSA, a gridded sensor array in southern New Mexico. The underlying goal of this endeavor is to improve atmospheric models (and sensor technology), by optimizing high-resolution forecast-observation comparisons. A description of the MSA vision and evolving program follows.

### 1.2.1 MSA Vision

ARL’s vision for the MSA Program was built, in part, on addressing the national need for high-resolution observational data in developing and evaluating high-resolution micro to mesogamma-scale forecast models. Such model development includes, but is not limited to, exploring new phenomena and analysis methods for near surface, high-resolution weather forecasting. The MSA model evaluation interests are aimed at integrating a powerful observation resource with both traditional and nontraditional model validation and verification (V&V) methods. Examples of both V&V methods are in Section 9.

The 3 underlying objectives framing the MSA Program are the following: 1) to provide reliable and persistent data resources that allow atmospheric modelers and sensor developers to validate and compare model and sensor performance with meteorological observations at and near the surface, in terrain of varying complexity; 2) to significantly improve high-resolution atmospheric models in the boundary layer; and 3) to improve model and sensor accuracy and efficiency.

A description of this multiphase MSA Program is described in the next section.

### 1.2.2 MSA Program Phases

The ARL MSA Program was subdivided into several phases. MSA-Phase I was designated as a “Proof of Concept”. This “Proof of Concept” consisted of 5 meteorological towers, which acquired 8 weeks of 24 h/day–7 days/week (24/7) data, over southern New Mexico. At the time of this writing, Phase I was actively working through its post-field campaign tasks. Sections 3 through 7 will describe Phase I, in more detail.

MSA-Phase II is envisioned as a 36-tower array located in a New Mexico desert valley. Phase III supplements the Phase II array with another 36-tower array located climatologically upwind from the New Mexico desert valley. Figure 1 shows projected locations for the MSA Phases II and III. A taller than 2-km mountain range would separate the two 36-tower data resources. The Phase III configuration is also envisioned to have additional sensor types, such as tethersondes, rawinsondes, and possibly unmanned aerial systems.

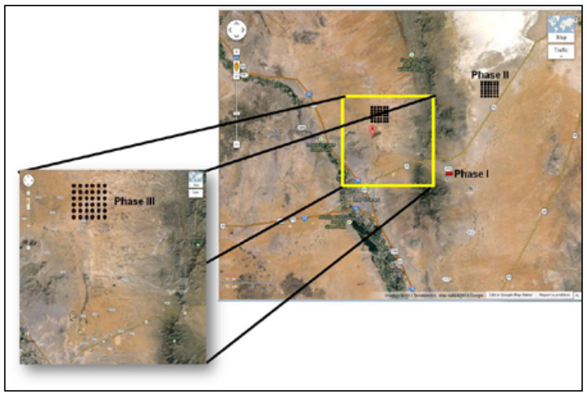


Fig. 1 MSA locations—Phase I field campaign was completed in 2014, the locations for Phases II and III are projected locations

Phase IV shifts the focus onto mobilizing the array. A portable two-dimensional array would be created and include a triple Lidar. The Lidar would be coordinated with sodars, tethersondes, and an unmanned aerial vehicle. Phase V is slated as a remote test site capability to be coupled with other field campaigns.

# 2. Sensor Calibration

The sensor calibration task was divided into 2 categories: dynamic and thermodynamic sensors. The dynamic sensor consisted of 20 ultrasonic anemometers measuring winds (see Fig. 2), sonic-temperature and speed of sound. The thermodynamic sensor group included 26 sensors representing 4 variables: pressure, temperature, relative humidity, and insolation.

Timely access to a calibration laboratory was not an option; consequently, a side-by-side sensor intercomparison (a relative calibration) was conducted. The configuration and preliminary results are described next.

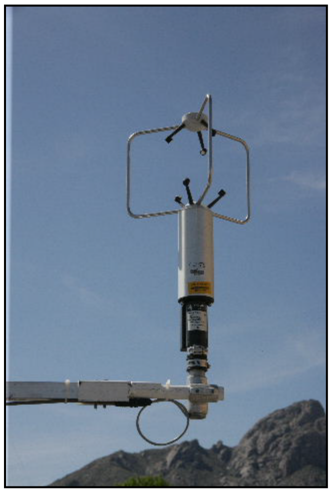


Fig. 2 The RM Young 81000 Ultrasonic Anemometer was the sensor quantifying the dynamic atmospheric attributes

## 2.1 Dynamic Sensor Calibration

The sonic anemometer intercomparisons were conducted on the flat roof of a 2 story building. The anemometers were arranged in a row, mounted on 4 tripods with 2 sensors per tripod (see Fig. 3). The sensors on each tripod were mounted at the ends of a crossbar, with all 8 sensors separated by an even spacing of 1.13 m. This pattern was repeated for all 3 anemometer calibration periods keeping 1 sensor common to all 3 acquisition periods. A total of 20 sonic anemometers were compared (see Table 1), using a north-south orientation; prevailing winds were from the west.

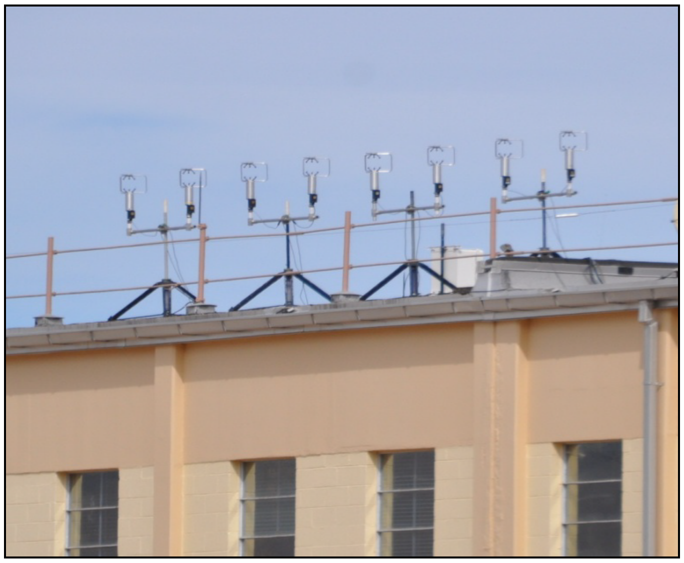


Fig. 3 Dynamic sensor side-by-side intercomparisons

The sonic measurements were acquired on a central data collection computer, using an 8-port RS232 adapter and Labview software to collect and timestamp each reading.

Table 1 Pre-Phase I field-campaign sonic calibration position assignments. Position number 1 was the southern-most position—8 northern-most. Each number listed under a Group heading represents a specific sonic.

| **Sonic Calibration Sampling Positions** | **Group I** (2014 Feb 10–13) | **Group II** (2014 Feb 14–17) | **Group III** (2014 Feb 19–23) |
|---|---|---|---|
| 1 | #1343 | #626 | #498 |
| 2 | #1355 | #633 | #499 |
| 3 | #1356 | #634 | #633 |
| 4 | #1341 | #1341 | #1341 |
| 5 | #1357 | #637 | #646 |
| 6 | #1359 | #638 | #650 |
| 7 | #1361 | #646 | #712 |
| 8 | #1370 | #1354 | #726 |

A qualitative review of the side-by-side data showed the sensors as being worthy of the MSA "Proof of Concept" data acquisition task. A more detailed analysis will be reported in a separate document.

## 2.2 Thermodynamic Sensor Calibration

Side-by-side, thermodynamic sensor intercomparisons were executed prior to erecting the MSA Phase I "Proof of Concept" towers. The intercomparison was situated on the south side of a 1-story office building, and consisted of 26 sensors, linked by 5 Campbell data logger systems (see Fig. 4). The thermodynamic variables sampled included pressure, temperature, relative humidity, and solar radiation. All thermodynamic data were saved in 1-min averaged samples.

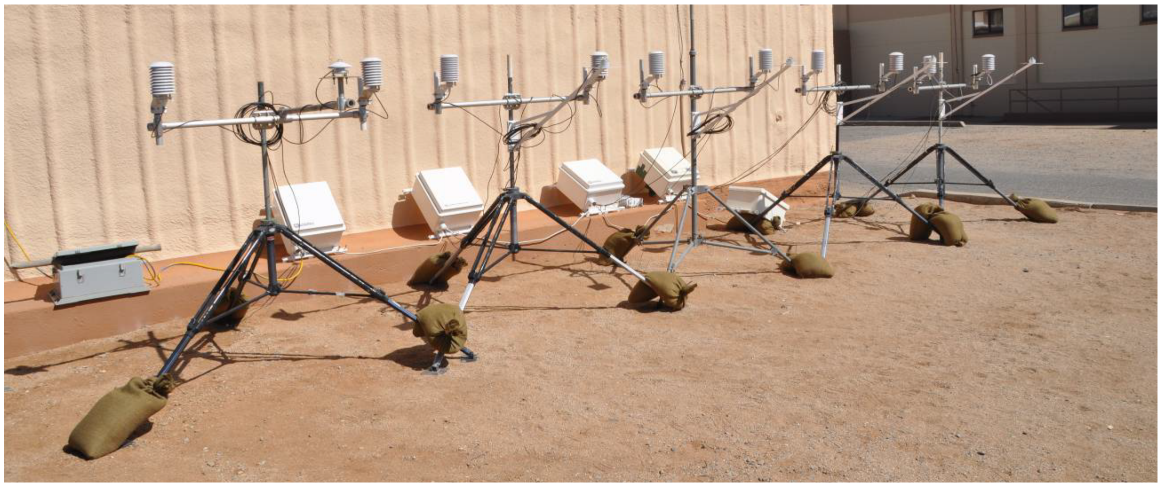


Fig. 4 Thermodynamic sensor side-by-side intercomparisons

The center tripod had new, or calibrated within the last year, sensors and a CR23X mounted on it. Sensors mounted on tripods to either side of the recently calibrated “standard”, had the “unknowns” or instruments for testing.

All sensor positions on the individual tripods (height above ground, distance from the tripod or boom) were carefully aligned to within 2 cm of each other. For reference, the wall of the building, which was north of the tripods, was aligned almost exactly on a true east-west line.

The MSA Phase I “Proof of Concept” used hardware components from previous field tests. This resource insured that the components had a proven durability and that system development costs would be kept very low.

Table 2 provides a brief description of the thermodynamic and dynamic components used for the MSA “Proof of Concept” field campaign.

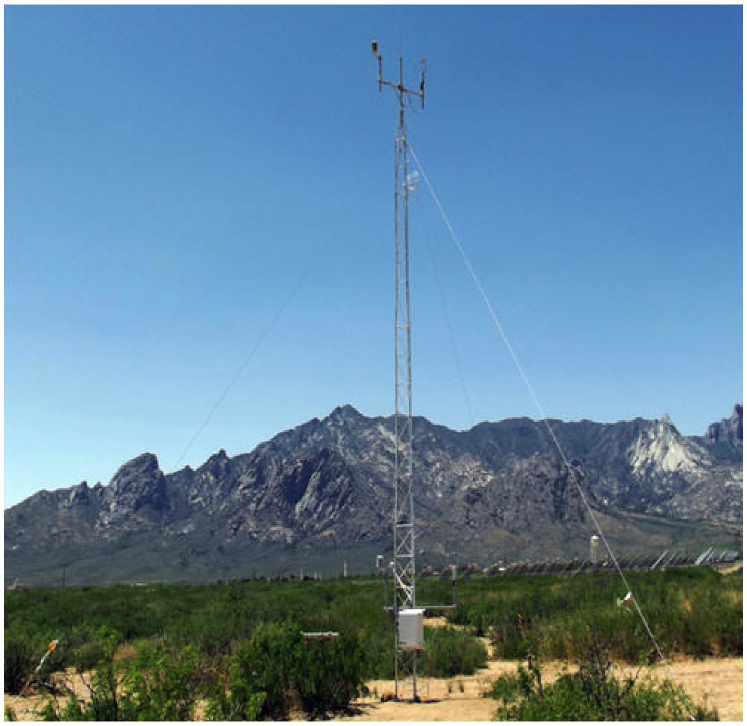
Table 2 Sensors used in the MSA Phase I "Proof of Concept" design

| Variable | Sensor | Manufacturer | Model | Units |
|---|---|---|---|---|
| Pressure | Barometer | Vaisala | PTB-101B/PTB110 | Millibars |
| Temperature | Thermometer | Campbell | T107 | Celsius |
| Temperature/ Relative Humidity | Thermometer/ Hygrometer | Rotronics | HC2S3 | Celsius/ Percent |
| Solar Radiation | Pyranometer | Kipp/Zonen | CM3/CMP3 | W/m$^2$ |
| Micrologger | ALL | Campbell Scientific | CR23X | . . . |
| Wind Speed and Wind Direction | Sonic Anemometer | RM Young | 81000 | m/s and degrees |

Relative calibration data were acquired from 2014 February 25 to March 06. A qualitative review of the side-by-side data showed the sensors as being worthy of the MSA "Proof of Concept" data acquisition. A more detailed description of the software used and a data analysis will be reported in a separate document.

## 3. Tower Design Description

The MSA Phase I 10-m tower was a portable lightweight aluminum tower from Climatronics Corporation that consisted of three 10-ft (3.048-m) sections bolted together on each vertical leg of the tower (see Fig. 5a and b). The tower included a tilt-down base plate and adjustable center mast capable of extending the height to accommodate a 10-m sensor level and a lightning rod (at the top) with a full height grounding kit. Each vertical leg of the tower was guy-wired to the ground from near the top, at approximately 9 m.



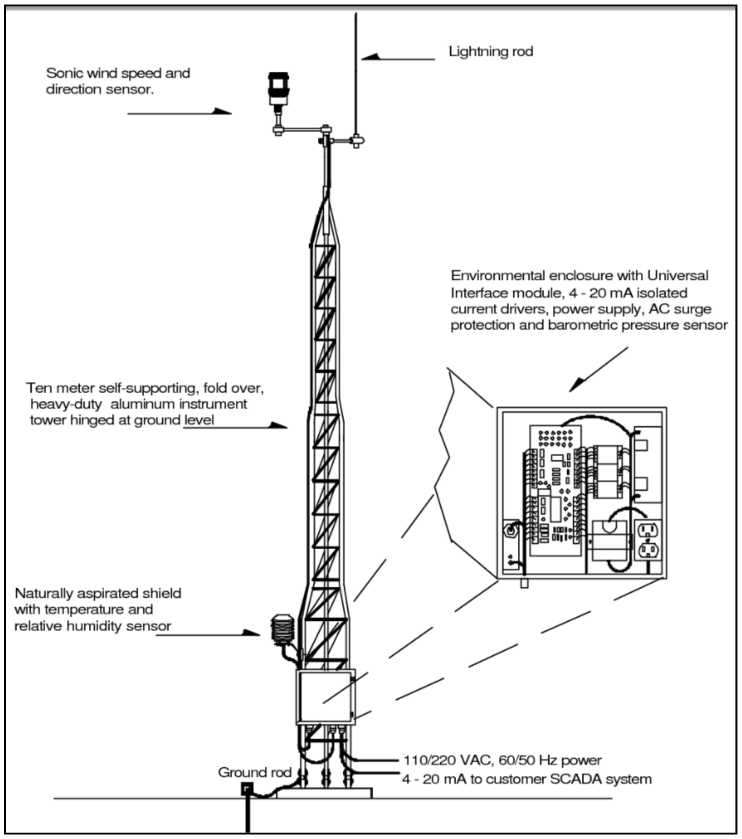

Lightning rod
Sonic wind speed and direction sensor.
Environmental enclosure with Universal Interface module, 4 - 20 mA isolated current drivers, power supply, AC surge protection and barometric pressure sensor
Ten meter self-supporting, fold over, heavy-duty aluminum instrument tower hinged at ground level
Naturally aspirated shield with temperature and relative humidity sensor
110/220 VAC, 60/50 Hz power
4 - 20 mA to customer SCADA system
Ground rod

Fig. 5 a) MSA tower #4 at White Sands Missile Range (WSMR); NM, b) 10-m tower (Climatronics Corp 2014). Fig. 5b used with permission from David W Gilmore Met One Instruments, Inc. – Climatronics Division.

There were 2 horizontal boom arms (1.25-inch outside diameter thick wall aluminum pipe) mounted, (true) east to west, on the towers to accommodate meteorological sensors at the 2- and 10-m levels. The bottom boom arm was mounted at approximately 1.33 m. Risers were used to place the sensors at precisely 2-m above ground level (AGL). A sonic anemometer (RMYoung 81000) was mounted on the climatologically upwind (west) end of the bottom boom arm 1.143-m (45 inches) away from the tower framework. A (Rotronics HC3S2) temperature/relative humidity sensor was mounted inside a naturally aspirated temperature shield on the east end of the bottom boom arm. The upper boom arm was mounted on the tower center mast and risers were used to place the sensors at precisely 10-m AGL. A second (RMYoung 81000) sonic anemometer was mounted on the west end of the upper boom and a (Campbell Scientific T107) temperature probe was mounted on the east end. A Kipp & Zonen CM3 pyranometer was mounted, with a custom made boom arm, on the south side of the tower and was attached at a level so that the custom made boom arm placed the sensor at precisely 2-m AGL. The final meteorological sensor, a (Vaisala PTB 101, PTB 110 on tower 1) digital barometer, was mounted inside the environmental enclosure near the bottom of the tower. The pressure sensor was mounted between 1- and 1.5-m AGL, and was present at Towers 1, 2, and 4 only.

The white environmental enclosure was a Campbell ENC 16/18 unit fiberglass-reinforced polyester box (width × height × depth: 16 inches × 18 inches × 9 inches) mounted on the north-east side of the tower at approximately 0.75-m AGL (see Fig. 6). When the field campaign was extended, these enclosures were reconfigured with vents and had a fan installed to prevent overheating.

The enclosure box contained the barometer, system data logger, a laptop computer, an Edgeport multiport serial-to-Universal Serial Bus (USB) convertor, relocatable power tap, and computer-to-antenna electronics. Tower #3 had a TM 1000A global positioning system (GPS) network time server, as well as, the electronics that supported the relay antenna linking the tower array to the computer base station located several miles away.

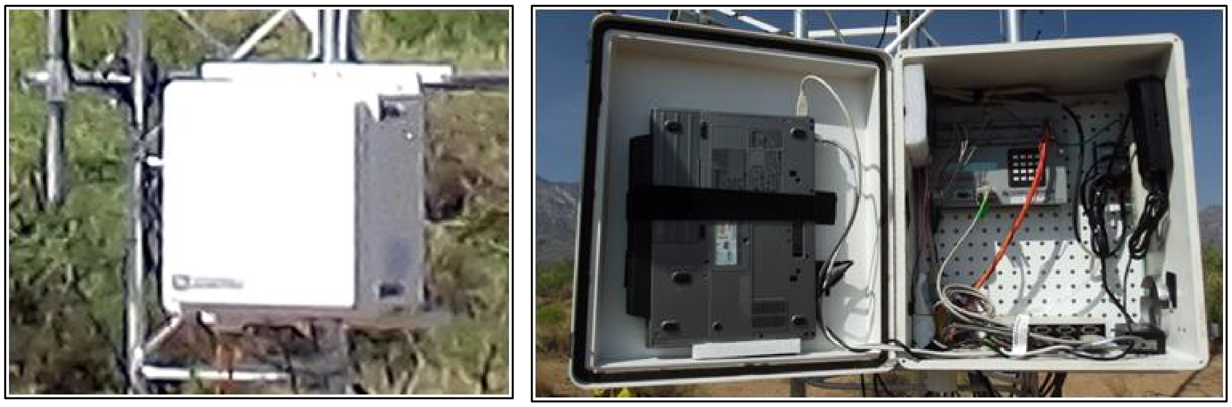


Fig. 6 MSA tower environmental enclosure

The tower instrumentation was powered with both 12-V direct current (DC) and 110-V alternating current (AC) current. The 12-V batteries were mounted on a wooden stand, away from the tower that also served as the mount for a 60 cell, 250-W SHARP (Sharp model ND250QCS 250W) solar photovoltaic (PV) panel. The PV panel charged the batteries during the day. A Cotek S300-112 Pure Sine Wave l DC to AC power inverter was used to supply the AC current (see Fig. 7). As part of a Phase Ib, the 12-V battery pair at each tower was later replaced by four 6-V Interstate GC2-XHD-UT batteries. The PV batteries were 232-ampere-hour (Ah) rated and were used to increase the stability of the daytime power delivery.

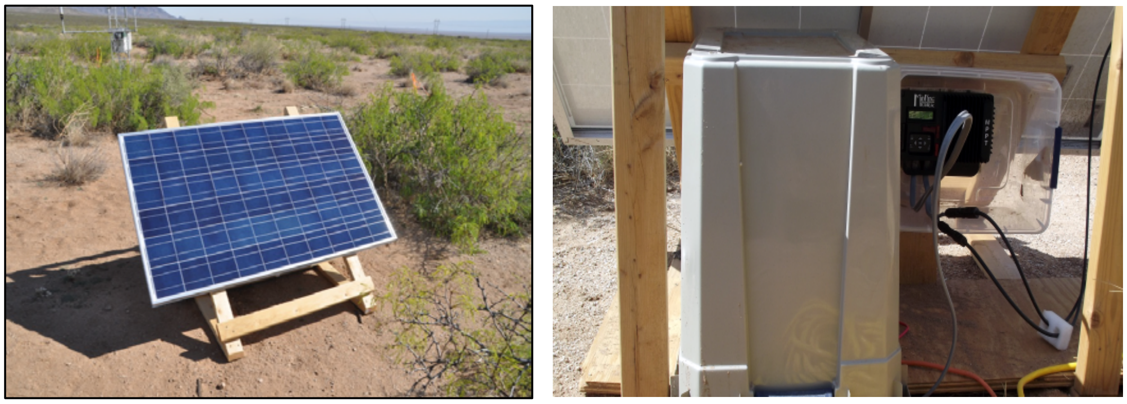


Fig. 7 MSA tower solar panel, batteries (covered), and power inverter

The communication links between towers are described in Section 6.

## 4. Tower Data Acquisition Systems (DAS)

Two DAS were used for the MSA Phase I field campaign: a thermodynamic DAS (“Logger DAS”) and a dynamic DAS (“Sonic DAS”). Figure 8 shows the 2 systems and their integration data flow pattern. Each DAS is described separately.

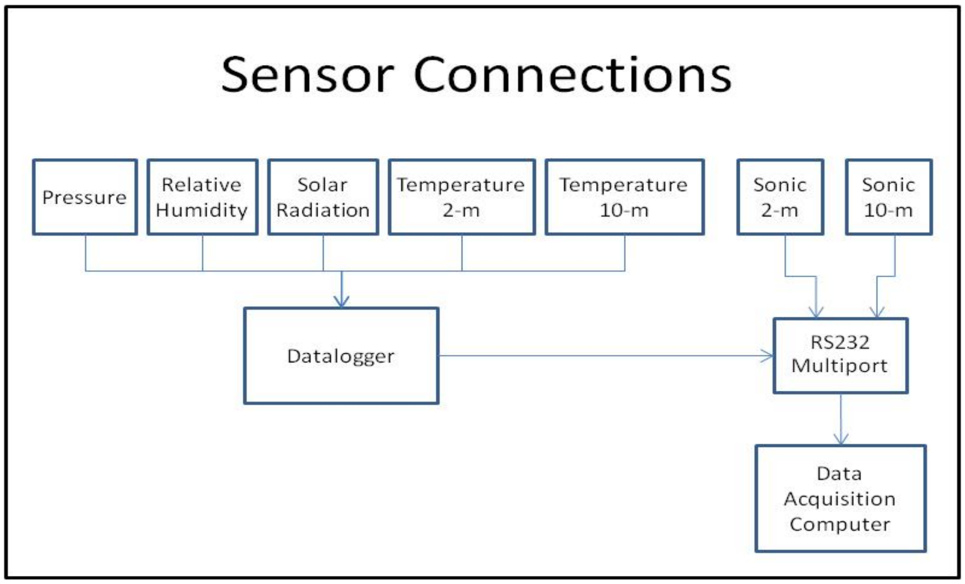


Fig. 8 The integrated thermodynamic and dynamic DAS

## 4.1 Thermodynamic (Logger) DAS

The thermodynamic DAS collected measurements of pressure, temperature, relative humidity and insolation. These data were linked together using a micrologger (see Fig. 6); thus, this part of the configuration was also called "Logger DAS". All thermodynamic sensors were hardwired into a Campbell Scientific, CR23X Micrologger located in the all-weather enclosure, mounted on the tower. When this data logger was connected to a Microsoft Windows computer running Loggernet software, we were able to manipulate the program in the CR23X Micrologger and do near real-time quality control data screening. When this data-logger was wired to the MSA computer, via an RS-232 to USB using the Digi Edgeport RS232 multiport adapter, control of the logger output was handed off to the MSA computer.

Prior to the sensor installation on the towers, the towers were laid out to identify required heights, fasteners, grounding cables, cable tie-downs, data cables, tower cross-arms, and instrument placing. Each sensor was individually calibrated and the CR23X Micrologger software was tested (see Section 2).

The Campbell Scientific CR23X Micrologger program that was created used Loggernet software-Version 3.4.1 for Microsoft Windows. This program was downloaded into the CR23X Micrologger via a USB to RS-232 cable. The program's function was to control the data collection and distribution. Specifically, the program sampled atmospheric conditions every 10 s, then output 1-min averages. Appendix A displays a sample of the micrologger program (Campbell Scientific 2004).

### 4.2 Dynamic (Sonic) DAS

The dynamic DAS collected all the ultrasonic anemometer data and was also referred to as the “Sonic DAS”. The serial data collection for each tower included both RM Young 81000 ultrasonic anemometers. The collection design also downloaded data from the Campbell Scientific CR23X data logger.

These data were collated by a laptop computer, which managed collection programs, time synchronization, communications, and file management. The computer was connected to a Digi Edgeport RS232 multiport adapter, allowing up to 8 serial ports. The data from each port was collected using a C program, which polled the serial ports and saved each data line with a prefixed time stamp. Each serial data source was saved to its own file, and a new file was spawned at the start of each hour. After a new file was created, the files from the previous hour were automatically averaged by another C program.

The system clock on each tower was synchronized using Network Time Protocol (NTP), to ensure all time stamps from each tower were consistent.

## 5. Tower Power Design

The MSA towers needed to operate independently, in remote locations without an infrastructure. The primary challenge was providing reliable electrical power. The challenge was answered with solar power, after assessing the system power requirements:

> DC systems include a logger and 2 sonics, which required approximately 5 W. The AC systems included wireless technology (4.5 W) and a laptop. The average laptop power need varied by unit, with a typical draw between 15 and 35 W. A maximum average requirement was determined to be about 45 W, over 24 h; meaning 1,080 watt-hours (whr) per day. With an average sunlight of 7 h, an average of 150 W was required.

This total wattage was more than the typical portable solar panels output could provide. Consequently, a grid tie sized panel (a 60-cell panel, sized for residential/commercial use) was selected, with an output of 250 W. A secondary benefit to choosing a grid tie sized module was their lower price; grid tie sized panels are mass produced for use by commercial solar power PV arrays.

The use of the 60-cell panel necessitated the use of a maximum power point tracking (MPPT) charge controller, which allowed for a wide difference between input and output voltages. A Morningstar 45 amp controller was selected for durability and simplicity. A pure sine wave inverter was selected to prevent interference with active power factor circuits in computer power supplies. Figure 9 summarizes the MSA-Phase I power connections.

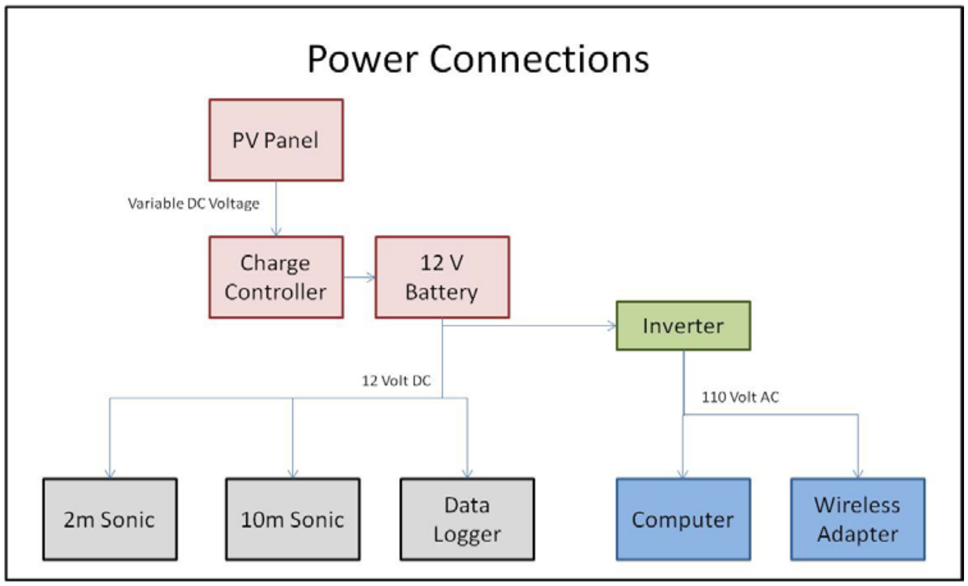


Fig. 9   Summary of the MSA-Phase I power connections

## 6. Communications

MSA communications served 3 functions: They downloaded the tower data, allowed system maintenance and monitoring, and they provided a conduit for time synchronization. During the initial Phase I field campaign, procurement delays of the wireless technology resulted in a daily manual download of data. The 5-tower data download required a total of approximately 800 MB per day per tower. While the manual daily data download was successfully executed from the 5 towers, this method did not represent a practical mode for the larger full-scale (36 towers) design. Once the wireless communication technology was installed, the daily quality-control assessments were executed much more efficiently. Visual tower inspections were reduced to 1 per week during the field campaign time period.

The wireless communications design was composed of wireless adapters mounted on each tower. On Tower 2, an additional large omnidirectional antenna established the central wireless adapter for the tower network. This adapter was setup in “access point” mode, whereas, the other tower adapters were setup in “client mode”. The “client mode” adapters used directional antennas to link with the central access point. A backhaul link to a central control station (MSA Headquarters) was established using directional adapters in “bridge mode”. These adapters were placed on Tower 3 and the roof of the central MSA control station building. This backhaul link was connected to the “client adapter” at Tower 3 using an Ethernet switch. Tower 3 was chosen for the backhaul link due to its clear line of sight. A NTP service was added to the data flow using GPS time servers located at Tower 4 and in the central control station. Automated data download was provided using secure copy through cron scripts to download all data to the central control station building every hour. Figure 10 diagrams the wireless design.

Networking the data acquisition computers facilitated the monitoring of current data collection and allowed updates for any software that may need modification.

The presence of a network allowed automatic synchronization of all system clocks using NTP, which was built into the Linux operating system. This synchronization was critical for maintaining consistent time stamps across the data array.

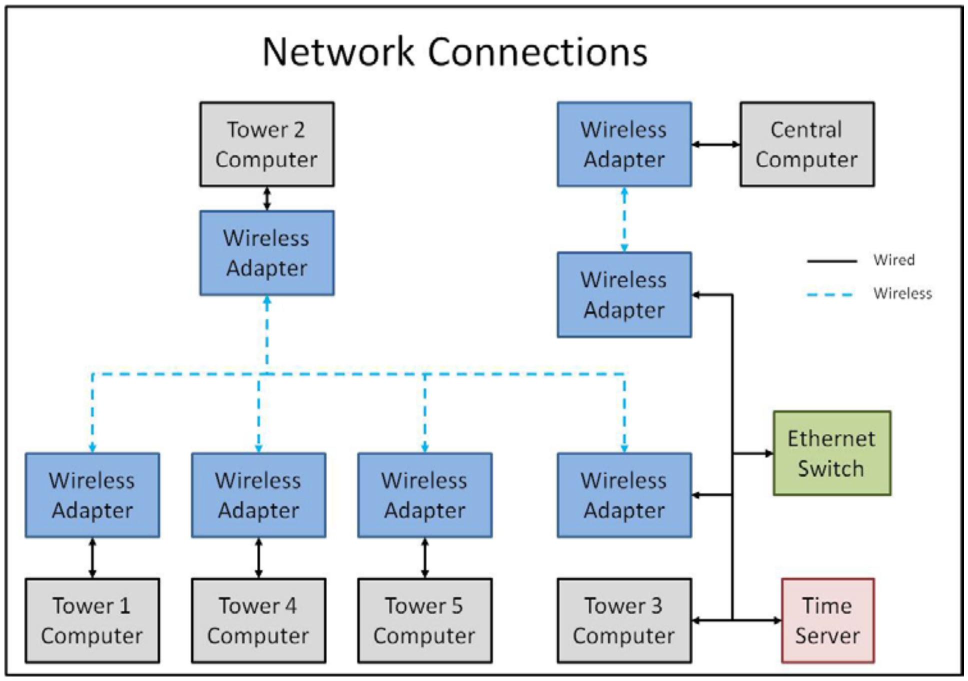


Fig. 10 MSA-Phase I communications/network connections

## 7. Data Processing

The MSA data processing began with raw data being acquired by the sensors. These raw data were preserved with their respective time stamps and formatting. The raw sampling rates were 1-min averages for the Logger DAS data, and 20-Hz data for the Sonic DAS data. Examples of the computer programs executing the data processing functions are in Appendixes B–D.

The Sonic raw data files were reduced to 1-min averages and merged with the Logger data into 1 file. Prior to the wireless communications, the merged data were manually downloaded from each tower to a portable storage medium. Using the “sneaker net”, these files were transferred to a UNIX workstation that served as the MSA hub at the MSA control station building. With the implementation of the wireless network, a cron job was created to run every hour on the UNIX workstation to automatically download merged data from each tower to the MSA hub.

An MSA data visualization tool was used to process the merged data into plots representing the last 24–48-h period, in Mountain Standard Time (MST). The MST time stamp was locally chosen for its efficient daily data assessment, based on the local diurnal effects. The automated plotting routine consisted of 2 programs: The wrapper program was written in Visual Basic Script, and merged all the hourly files into one 24-h period file, for each tower. The 24-h data file was then passed to a Graphic Layout Engine (GLE) script, to plot the data. The output generated a summary page with 12 parameter plots for each tower. A wrapper program saved the tower plots as a jpg image, giving it a descriptive name. See Appendix E, for the Visual Basic Script source code, and Appendix F, for the GLE program.

To support the MSA model V&V application project, another data processing tool reformatted MSA merged data into Model Evaluation Tools (MET) specific American Standard Code for Information Interchange (ASCII) Point Observation text files (See Appendices G and H). The V&V application took merged 24-h data files of all towers, extracted certain parameters, calculated additional variables and reformatted the results into a configuration compatible with the V&V assessment software-MET Point-Stat. A log was then kept of the reformatted files, for a given date, that were transferred to the destination workstation for V&V processing.

An effort to automate the above processes is underway. A graphical user interface (GUI) is being constructed using WinBatch, which will automate the File Transfer Protocol procedure from the field computer to a V&V workstation, as well as, the Met ASCII reformatting process.

# 8. Automated Quality Assurance Protocol

During the MSA "Proof of Concept" field campaign, a daily data review was conducted. The process began with the MSA data from the previous 36 h being converted from data files into plotted time series of each sampled variable, as well as some calculated parameters (see Section 7). These time series were visually reviewed on a daily basis. The analyst reviewed the plots for sonic and logger errors, tower battery voltage dropping too low, DAS panel temperature rising too high, etc. An inter-tower data comparison was also assessed. The analyst then created a daily report that gave the status of all towers and detailed any anomalies observed over the previous 36 h along with weather conditions over this time period and any recommendations. Tools to execute the above tasks were tested, refined and re-tested throughout this daily data processing implementation exercise.

Investigating future MSA data processing tools was part of the MSA Phase I "Proof of Concept" effort. One area investigated was the automating of the data Quality Assurance (QA) Protocol. The remainder of this section describes the key findings.

Invalid meteorological measurements can be caused by equipment failure, a lightning strike, or a power outage. The purpose of an automated QA algorithm is to flag data that fall outside designated ranges, so that the MSA team is alerted early to potential problems. The QA limits are designed to provide an early warning of sensor failure, so that repairs can be made quickly with a minimum of data loss. The following QA limits are modified from the protocols used by the Oklahoma Mesonet, as discussed in Fiebrich et al. (2010).

## 8.1 QA Limits

A scenario for utilizing QA limits would be the following. An automated QA would flag data overnight. The MSA team would review the flagged data every morning and determine what corrective actions, if any, need to be taken. The team would also review daily time series plots of each meteorological parameter from each tower. Plots of voltage output and panel temperature would be reviewed. Both a manual review and an automated QA Protocol would be an integral part of the quality assurance process in order to achieve the goal of producing high-quality meteorological data.

Initial instrument parameters would be set according to the manufacturer specifications (see the following examples).

### 8.1.1 Battery Voltage

The normal range for field batteries is 10 to 16 V. The following voltages would be flagged to ensure that battery problems are corrected before data loss occurs:

- Data collected with a battery voltage of 11.5 V is likely to be valid.
- Battery voltages below 11.5 V indicate that the battery may soon fail, so voltage measurements below 11.5 V would be flagged.
- When the solar radiation is at its peak, the battery voltage should also be high. A “peak” battery voltage below 13.5 V would be flagged, because it would indicate a decreased efficiency in the solar energy collection process.

### 8.1.2 Panel Temperature

Logger panel temperatures less than 0 °C, or greater than 40 °C, would be flagged, because computers and other electronic equipment do not operate properly at these temperatures. For the area of the MSA Phase I field campaign site, as well as the proposed Phase II and III sites, such extreme temperature are part of the climatological record. The lowest temperature reported was −23 °C in January of 1962, and the highest temperature was 43 °C in June of 1994. Heat released by the computer and the sensors can be about 5 °C warmer than the ambient temperature. Combining the local ambient warmth with the equipment-generated heat, there is a high likelihood that the data logger operating ranges will be exceeded during the hot summer months.

### 8.1.3 Internal Quality Control

Sensors that would be installed on the towers have internal components that monitor the internal temperature. When the internal temperature of a temperature sensor falls outside of the recommended operating range, the instrument QA would automatically flag the data.

### 8.1.4 Climatological Average Values

Climatological minimum and maximum values were available for all parameters used in Phase I, except relative humidity. The MSA quality assurance algorithm would flag measurements that are outside of the climatological minimum and maximum averages. Because extreme weather events can cause meteorological measurements to fall outside climatological norms, the MSA team would determine whether flagged data are valid.

### 8.1.5 Air Temperature (2- and 10-m AGL)

The climatological average, minimum and maximum data temperature in Table 3 are adapted from the Weather Channel (2014), and covers a period of approximately fifty years over an area near the MSA Phase I site. The specific location of these measurements was not available from the Weather Channel.

Table 3 Climatological temperatures in degrees Centigrade, at WSMR, NM. Adapted from Weather Channel at http://www.weather.com/weather/wxclimatology/monthly/88002.

| Month | Average High | Average Low | Average | Record High | Year | Record Low | Year |
|---|---|---|---|---|---|---|---|
| January | 15 | –2 | 7 | 26 | 1970 | –23 | 1962 |
| February | 18 | 1 | 9 | 29 | 1986 | –21 | 2011 |
| March | 21 | 3 | 12 | 32 | 1989 | –12 | 1965 |
| April | 26 | 7 | 17 | 36 | 2000 | –4 | 1973 |
| May | 31 | 12 | 22 | 40 | 2005 | –3 | 1967 |
| June | 35 | 17 | 26 | 43 | 1994 | 6 | 1971 |
| July | 35 | 20 | 28 | 42 | 1994 | 13 | 1983 |
| August | 33 | 19 | 27 | 40 | 2007 | 11 | 1970 |
| September | 31 | 16 | 23 | 39 | 2011 | 4 | 1965 |
| October | 26 | 8 | 17 | 35 | 2000 | –6 | 1970 |
| November | 19 | 2 | 11 | 31 | 1988 | –20 | 1976 |
| December | 14 | –2 | 7 | 25 | 1987 | –15 | 1987 |

### 8.1.6 Relative Humidity (2-m AGL)

The average relative humidity values in Table 4 were obtained from the White Sands Missile Range Climate Calendar (Fugate and Chambers 1972). From 1948 to 1971, Fugate and Chambers tabulated average daily and monthly meteorological values for “Station A”, which was located at the base headquarters.

Table 4   The 1948 to 1971 average relative humidity at WSMR from Fugate and Chambers (1972)

| Month | Average 5 AM | Average 11 AM | Average 5 PM | Average 11 PM | Average |
|---|---|---|---|---|---|
| January | 54 | 42 | 38 | 47 | 45 |
| February | 49 | 36 | 29 | 40 | 39 |
| March | 41 | 28 | 22 | 33 | 31 |
| April | 35 | 23 | 17 | 27 | 26 |
| May | 34 | 21 | 16 | 25 | 24 |
| June | 38 | 23 | 18 | 28 | 27 |
| July | 58 | 36 | 31 | 46 | 43 |
| August | 59 | 37 | 31 | 43 | 43 |
| September | 56 | 36 | 30 | 45 | 42 |
| October | 51 | 33 | 29 | 42 | 39 |
| November | 51 | 34 | 34 | 44 | 41 |
| December | 56 | 42 | 38 | 49 | 46 |

### 8.1.7 Air Pressure (Surface)

The mean surface pressure values in Table 5 were obtained from the 1948 to 1971 data of Fugate and Chambers (1972). Table 5 also contains average low- and high-surface pressures.

Table 5   The 1948 to 1971 average surface pressure in millibars, at the WSMR Station A, near the Headquarters Building, from Fugate and Chambers (1972)

| Month | Average | Highest | Lowest |
|---|---|---|---|
| January | 872.64 | 888.49 | 851.92 |
| February | 870.88 | 886.12 | 852.59 |
| March | 869.39 | 886.45 | 852.59 |
| April | 868.98 | 885.78 | 852.93 |
| May | 869.25 | 883.07 | 856.32 |
| June | 869.39 | 879.34 | 857.00 |
| July | 871.93 | 882.05 | 862.41 |
| August | 872.40 | 880.70 | 863.77 |
| September | 871.96 | 882.05 | 860.38 |
| October | 872.47 | 887.81 | 856.66 |
| November | 872.61 | 888.49 | 856.32 |
| December | 872.61 | 890.01 | 853.27 |

### 8.1.8 Incoming Solar Radiation (2-m AGL)

The incoming solar radiation was measured at the White Sands Museum, NM. Using measurements from 2011 to 2013, the average solar radiation was calculated by the month and hour (Table 6). The standard deviation by month and hour was also calculated (Table 7). Solar radiation measurements that differ by more than 1 or 2 standard deviations from the climatological average values should be flagged. Nighttime and early morning hours measurements should equal zero.

Table 6 Average solar radiation in W/m2 for each hour and month. These values are averages of 2011–2013 measurements made at the White Sands Museum. Data were obtained from the University of Utah, Department of Atmospheric Sciences Web site (University of Utah 2014). http://mesowest.utah.edu/cgi-bin/droman/mesomap.cgi?state=NM&rawsflag=3

| Hour | Jan. | Feb. | March | April | May | June | July | Aug. | Sept. | Oct. | Nov. | Dec. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 1 | 1 | 2 | 15 | 15 | 9 | 3 | 0 | 0 | 2 | 0 |
| 7 | 27 | 71 | 55 | 101 | 160 | 170 | 113 | 78 | 42 | 19 | 77 | 30 |
| 8 | 187 | 267 | 235 | 311 | 362 | 371 | 276 | 241 | 194 | 180 | 245 | 158 |
| 9 | 365 | 455 | 447 | 518 | 573 | 564 | 449 | 423 | 377 | 380 | 399 | 307 |
| 10 | 505 | 619 | 641 | 714 | 734 | 726 | 609 | 588 | 544 | 546 | 517 | 420 |
| 11 | 592 | 714 | 781 | 855 | 880 | 849 | 749 | 724 | 669 | 673 | 578 | 489 |
| 12 | 607 | 721 | 854 | 942 | 954 | 916 | 808 | 798 | 730 | 731 | 594 | 506 |
| 13 | 565 | 686 | 826 | 940 | 943 | 934 | 802 | 783 | 747 | 731 | 558 | 467 |
| 14 | 483 | 602 | 769 | 890 | 883 | 876 | 750 | 731 | 700 | 663 | 448 | 375 |
| 15 | 340 | 437 | 640 | 778 | 810 | 787 | 659 | 617 | 586 | 535 | 294 | 237 |
| 16 | 159 | 270 | 480 | 633 | 656 | 627 | 527 | 514 | 445 | 386 | 116 | 73 |
| 17 | 11 | 73 | 288 | 421 | 475 | 459 | 363 | 351 | 274 | 207 | 18 | 3 |
| 18 | 0 | 1 | 115 | 216 | 272 | 275 | 211 | 178 | 108 | 31 | 1 | 0 |
| 19 | 0 | 0 | 10 | 41 | 88 | 109 | 76 | 41 | 7 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 3 | 8 | 5 | 1 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 7 Standard deviation of solar radiation in W/m2 for each hour and month. These 2011-2013 measurements made at the White Sands Museum. Data were obtained from the University of Utah, Department of Atmospheric Sciences Web site. http://mesowest.utah.edu/cgi-bin/droman/mesomap.cgi?state=NM&rawsflag=3

| Hour | Jan. | Feb. | March | April | May | June | July | Aug. | Sept. | Oct. | Nov. | Dec. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 2 | 4 | 5 | 19 | 13 | 10 | 4 | 1 | 0 | 4 | 0 |
| 7 | 31 | 60 | 65 | 66 | 69 | 64 | 63 | 52 | 41 | 26 | 64 | 36 |
| 8 | 77 | 101 | 118 | 91 | 106 | 68 | 99 | 96 | 194 | 70 | 104 | 89 |
| 9 | 106 | 128 | 139 | 121 | 128 | 93 | 142 | 122 | 132 | 84 | 137 | 136 |
| 10 | 133 | 138 | 143 | 127 | 135 | 92 | 155 | 137 | 153 | 102 | 162 | 167 |
| 11 | 143 | 153 | 140 | 152 | 146 | 116 | 164 | 161 | 185 | 99 | 173 | 185 |
| 12 | 146 | 184 | 137 | 129 | 155 | 143 | 191 | 171 | 209 | 113 | 166 | 178 |
| 13 | 149 | 166 | 182 | 147 | 181 | 119 | 221 | 209 | 212 | 123 | 144 | 161 |
| 14 | 129 | 155 | 193 | 138 | 172 | 114 | 235 | 226 | 204 | 129 | 142 | 137 |
| 15 | 113 | 140 | 179 | 124 | 144 | 145 | 250 | 229 | 184 | 133 | 130 | 107 |
| 16 | 87 | 104 | 179 | 118 | 143 | 170 | 223 | 197 | 160 | 108 | 108 | 67 |
| 17 | 12 | 68 | 161 | 115 | 121 | 142 | 186 | 156 | 121 | 88 | 49 | 4 |
| 18 | 0 | 3 | 101 | 82 | 93 | 104 | 118 | 103 | 74 | 43 | 4 | 0 |
| 19 | 0 | 0 | 18 | 43 | 56 | 59 | 58 | 43 | 12 | 1 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 5 | 11 | 8 | 2 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 8.2 Redundant Data

Meteorological measurements that show no change over time, or change too rapidly could be indicating that there is a sensor malfunction or other problem occurring. The automated QA system would be designed to flag data that fall outside of temporal benchmarks so that the MSA team could quickly attend to equipment issues and minimize data loss. Note that temporal benchmarks refer to a rate at which meteorological measurements vary over a given time period.

The following temporal benchmarks were adapted from Fiebrich et al. (2010). Measurements that fall outside these quality assurance limits would be flagged in a future MSA data processing routine. Note that because a rapidly moving front can also cause rapid changes in temperature, pressure and solar radiation, the MSA team would need to determine whether the flagged data are valid or a result of a sensor malfunction.

### 8.2.1 Air Temperature (2- and 10-m AGL)

- Temperatures increasing by more than 6 °C in 5 min.
- Temperatures decreasing by more than 8 °C in 5 min.
- Temperature data that do not change by more than 0.18 °C.

### 8.2.2 Relative Humidity (2-m AGL)

- Relative humidity values that increase more than 22% in 5 min.
- Relative humidity values that decrease more than 23% in 5 min.
- Relative humidity values that do not change at least 0.1% in 360 min.

### 8.2.3 Air Pressure (Surface)

- Surface pressures that increase more than 5 hPa in 5 min.
- Surface pressures that decrease more than 4 hPa in 5 min.
- Surface pressures that do not change more than 0.1 hPa in 30 min.

### 8.2.4 Solar Radiation

- Solar radiation measurements that change by more than 800 W/m$^2$.
- Solar radiation measurements that do not change at least 0.1 W/m$^2$.
- An adjustment needs to be made for nocturnal measurements which will equal zero W/m$^2$.

## 8.3 Spatial Consistency of Data

Spatial consistency of data is another QA attribute. Observations that are not consistent with neighboring stations would need to be flagged. In general, spatial comparisons are most successful in finding erroneous data when the boundary layer is well-mixed. Comparison of air temperature data among neighboring stations is best conducted with moderate winds that are greater than 0.4 m s$^{-1}$ (Fiebrich et al. 2010).

One spatial consistency algorithm calculates the standard deviation between stations, and flags data that differ by more than 2 standard deviations from nearby stations. Another method is the Barnes objective analysis, which computes an expected value by assigning an exponentially decreasing weight as distance between a station and its neighbor increases. Fiebrich et al. (2010) discussed other methods such as the optimal interpolation technique and the spatial regression test. A future project would be to test some of these methods to determine which method will work best for MSA data.

## 9. Data Applications

Several data applications were identified for MSA Phase I. While most of these applications were scheduled to have publishable results in the next fiscal year, we can present preliminary results on the V&V application in this section.

MSA-Phase I objectives included the goal of acquiring meteorological observations near the surface and in close proximity to complex terrain and to provide ground truth data sets to verify Army high-resolution atmospheric models. As explained earlier, Phase I sensors sampled traditional meteorological variables at levels that are typically generated by weather forecasting models output. The 2 atmospheric models considered by the MSA designers for testing model V&V methods were the Weather Running Estimate-Nowcast (WRE-N), which is a variant of the Weather Research and Forecasting (WRF) model, and the Three-Dimensional Wind Field (3DWF) diagnostic model. Both models create output values for the variables on a regularly-spaced horizontal grid.

MSA meteorological towers were purposefully placed at locations that coincided with the 3DWF 100-m resolution grid points. This design was to minimize the interpolation uncertainty when determining the modeled value, which corresponds to the location of the measured value. The 3DWF microscale model assimilates a measured wind speed value and the magnitude of the east-west (u) and north-south (v) wind components from one of the towers, then diagnostically calculates the corresponding values at all other grid points for the output. Unfortunately, 3DWF was not available during the field campaign. Consequently, WRE-N was used to investigate the feasibility of conducting V&V studies.

Assessing the WRE-N performance, whose minimum grid spacing is 1 km, was somewhat problematic with this 100-m grid design, because the possibility of having grid points coincidence was almost nonexistent. Furthermore, the typical domain sizes for the 1-km grid were of the order of 100 km × 100 km, which would require vast numbers of towers to provide coincident measurements.

Another challenge for the mesoscale model WRE-N was the sampling rate. The measurements considered appropriate for validating WRE-N were 15-min averaged values. These were considered to be representative of the model output values that were intended to characterize conditions in the entire grid cell volume at the specified valid time of the forecast. The vertical, above ground sensor locations on each tower were specifically chosen to coincide with those of the WRE-N forecast values, which were at 2- and 10-m AGL.

The WRE-N model output consisted of hourly forecast values of air temperature, dew point temperature and relative humidity at 2-m AGL, and wind speed, u-component and v-component wind speeds at 10-m AGL. The forecasts were generated over a 126-km × 126-km horizontal grid domain with spacing of 1 km centered over WSMR and were valid at the top of each hour. The WRE-N also generated forecasts over larger domains at lower resolutions in a triple-nested configuration; however, only the output for the innermost nest was used for Phase I.

The MSA surface meteorological observations consisted of u, v, and w wind components measured at 10-m AGL; at 2-m AGL, sensors measured the air temperature, relative humidity, solar radiation and atmospheric barometric pressure. The air temperature, relative humidity, wind measurements provided the needed observation data directly, but the dew point temperature and mean sea level pressure had to be derived from the sensor data using standard formulae.

The dew point formulae used were provided in the Meteorological Assimilation Data Ingest System (MADIS) Application Program Interface (API) software (MADIS 2014). These formulae were implemented into the MSA Data Processing system. Fifteen minute averaged values of the variables were generated from the merged data produced by the MSA Data Processing system. A user application reformatted the merged data into the format required by the V&V software used to calculate the model error statistics. This format was called the MET Specific ASCII Point Observation format (National Center for Atmospheric Research 2013). The text files in this format were generated from the data collected 15 min prior to the top of the hour and are valid at the top of the hour.

## 9.1 V&V Tools and Preliminary Results

Having the WRE-N model output and the MSA Observations in hand, the next step was to compare the 2 resources. The tool to do this comparison is described in the next section, followed by a future data analysis tool.

### 9.1.1 NCAR MET Point-Stat

The MET is a set of verification tools developed by the WRF Developmental Testbed Center (DTC) to help WRF Users assess and evaluate the performance of a model (National Center for Atmospheric Research, 2013). MET Point-Stat is the tool that performs traditional, grid-to-point model verification and generates error statistics such as Mean Error, Mean Absolute Error, and Root Mean Square Error. The MET Point-Stat input uses the forecast model output grid in Gridded Binary (GRIB) format and observations in NetCDF format that have been generated by converting the ASCII Point Observation formatted text files containing the MSA measurements. Point-Stat extracts the model value of each variable corresponding with the location of the observed or measured value using interpolation, because the WRE-N grid points are typically not located at the observing sites. Point-Stat calculates the difference for every matched forecast-observation pair and generates the error statistics aggregated over all such pairs at a specific time in the modeling domain; thus, MET addresses some of the spatial mismatch between the model and observation grid sizes.

MET Point-Stat can be used to calculate traditional grid-to-point verification statistics for WRE-N forecasts run for specific case study periods over the WSMR domain. These results would represent aggregated model error statistics. When the 3DWF model becomes available, its output could also be evaluated using Point-Stat.

The mismatch between model and observation grid sizes is still a concern. Some nontraditional methods, such as the object-oriented spatial and neighborhood techniques, are being investigated as potential future assessment tools. A future statistical visualization tool being considered is the Geographic Information System (GIS), which is described in the next Section.

### 9.1.2 GIS

Looking toward the future, high-resolution modeling requires more focused spatial and temporal verification over parts of the domain. With a GIS, researchers can now consider terrain type/slope and land use effects and other spatial and temporal variables as explanatory metrics in model assessments. GIS techniques, when coupled with high-resolution point and gridded observations sets, allow location-based approaches that can facilitate the discovery of spatial and temporal scales not sufficiently resolved by the model, such as the turbulence effects or mountain and lee waves. ARL has started a GIS analysis of the matched forecast-observation pairs that are generated in text format by MET Point-Stat.

Location-based error statistics can be discerned using the GIS analysis tools. These tools are capable of identifying and codifying “natural” meteorologically and geographically defined subdomains. They can also be used to develop the means for computing traditional statistics over these domains to reveal spatial and temporal trends in the performance of the models.

# 10. Summary and Final Comments

Army decisions can be improved only when the information on which the decisions are based is improved. Environmentally dependent decisions involving the atmosphere rely heavily on accurate atmospheric models. “Army-scale” atmospheric models include high-resolution ($\leq$1 km) models. Locating meteorological observations to validate these high-resolution atmospheric models is very difficult. The NRC recognized this critical technological gap after they reviewed the US Weather Research, and Researcher–to-Operations progress and priorities in 2009. Numerous NRC conclusions and recommendations produced by this forum centered on this void in the atmospheric research community.

ARL has responded to the National and Military need by proposing an observational data resource specifically designed to address the “Army-scale”, high-resolution atmospheric model validation and verification issues. The manifested solution is called the “Meteorological Sensor Array” (MSA).

The MSA vision is built on improving Army decisions by strengthening environmentally-dependent decision aids through validated high-resolution atmospheric models in the boundary layer. The MSA is intended to provide reliable and persistent data resources that allow atmospheric modelers and sensor developers to validate and compare model and sensor performance with atmospheric observations at and near the surface over or in close proximity to terrain of varying complexity.

The multiphase program was initiated with a “Proof of Concept”, which included 5 equally spaced meteorological towers around a large Solar PV Farm. Phase II will integrate Phase I (“Proof of Concept”) lessons learned, as it expands the array into a 36-tower gridded design. The project location of Phase II is projected to be a mid-valley, desert location. Phase III will mirror the 36-towers configuration in a climatologically-upwind location with a tall mountain range in between the two 36-tower arrays. Additional volume measurements will supplement the design. Phase IV will focus on mobilizing the array with supplemental sampling technology. Phase V is envisioned as having the capability of participating in remote test site field campaigns.

The MSA Phase I field campaign began with a calibration of all sensors slated for field usage. Section 2 briefly describes the side-by-side sensor intercomparison configuration and results. The MSA fielded sensors are listed in Tables 1 and 2.

The MSA Tower design included a portable lightweight aluminum tower, with sensors mounted on the 2- and 10-m levels. The DAS were divided into 2 categories: Thermodynamic and Dynamic DAS. A Campbell Scientific CR23X micrologger assimilated the Thermodynamic, 1-min averaged data. The variables consisted of pressure, temperature (2- and 10-m AGL), relative humidity (2-m AGL) and insolation (2-m AGL). The Dynamic data originated from 2 RM Young 81000 Ultrasonic anemomemters (2- and 10-m AGL) sampling at 20 Hz. The variables acquired were wind speed, wind direction, u-component, v-component, w-component, speed of sound and sonic-temperature. The raw dynamic data were preserved in files on a laptop computer then reduced to 1-min averages. The dynamic and thermodynamic 1-min averages were merged with the thermodynamic data. A Digi Edgeport RS232 multiport adapter, supporting 8 serial ports, bridged the 2 data resources. A system clock on each tower was synchronized using the NTP.

Electrical power for the remotely located towers came from the sun. Each tower was designed with a solar PV panel, charge controller and battery storage unit, specifically tailored to support the temporary Phase I tower configuration. The DC electronics received a direct feed from the battery, whereas, the AC requirements of the computer and wireless adapters (communications) used an inverter. Future Phases will have reduced electrical tower requirements and, therefore, a modified power resource design.

The arrival of the Phase I wireless technology was delayed due to bureaucratic approval processes. Consequently, the initial data flow required a daily “sneaker net” download of data from each tower, manual maintenance and monitoring of each tower, and a constant time synchronization check. Once the wireless technology could be installed, these 3 functions were done automatically, and/or through a remote access. The network configuration brought all data to a single tower, which provided a time server, Ethernet switch, and a wireless adapter link back to the MSA central computer.

Data processing began with the averaging and merging of thermodynamic and dynamic data. Plots of the time series of each sensor were created and reviewed, daily. During the few times technological concerns surfaced, the MSA group immediately addressed and resolved the issues.

Research into automating and improving data quality assurance were investigated. One suggested improvement was to program quality assurance limits that would automatically flag potential data issues. Tables framing these limits were given in Section 8. Automating the intercomparison of neighboring sensors was also suggested.

The MSA design was intended for multiple applications. For Phase I, one of the primary applications was V&V. As part of the MSA task, the method for executing a high-resolution model V&V was explored. The mesogamma and microscale models considered were the WRE-N and 3DWF models. The tower locations and spacing were based on the 3DWF model; however, the model was not available during the field campaign, so the V&V implementation was postponed. Instead, the MSA observations were weighed against a WRE-N model output. Format changes to accommodate both the WRE-N model attributes (spatial and temporal scales) and the MET statistical tools were developed. Reasonable numerical results were produced confirming the feasibility of a model grid to observation point comparison. Elaboration and interpretation of the V&V results were left for a separate report.

Two core goals for the MSA Phase I task were to create a MSA development plan and to manifest a “Proof of Concept” from which future MSA Phases could be built. This report sketches the development plan and documents the manifested MSA Phase I “Proof of Concept”. Based on success of Phase I, the authors look forward to future, fruitful phases.

## 11. References

Bennett DA, Hutchison K, Albers SC, Bornstein RD. Preliminary results from polar-orbiting satellite data assimilation into LAPS with application to mesoscale modeling of the San Francisco Bay Area. Preprints, 10th Conference on Satellite Meteorology and Oceanography; Long Beach, CA. Amer Meteor Soc. 2000; 118–121.

Campbell Scientific, Inc. CR23X micrologger, operator’s manual. Revision: 11/06. Logan, (UT): Cambell Scientific, Inc.; 2004. Also available at http://www.campbellsci.com/documents/manuals/retired/cr23x.pdf.

Climatronics Corporation Web site. 10-meter weather station with sonic anemometer and 4–20 mA output. Bohemia, (NY): [accessed 2014 August 27]. http://www.climatronics.com/pdf/draft_specs/11.pdf.

Fiebrich CA, Morgan CR, McCombs AG. Quality assurance procedures for mesoscale meteorological data. Journal of Atmospheric and Oceanic Technology. 2010;27(10):1565–1582.

Fugate GM, Chambers JA. White Sands Missile Range Climate Calendar. White Sands Missile Range (NM): Atmospheric Science Laboratory (US); April 1972. Also available at http://www.dtic.mil/dtic/tr/fulltext/u2/743842.pdf.

MADIS Meteorological Assimilation Data Ingest System: National Oceanic and Atmospheric Administration/Earth Systems Research Laboratory (US). Headquarters, Silver Springs, (MD); 27 November 2013 [accessed 27 August 2014]. http://madis.noaa.gov.

National Center for Atmospheric Research (NCAR) (US): Developmental Testbed Center (US). Model evaluation tools user’s guide. Ver. 4.1 (METv4.1). Boulder (CO); 2013.

National Research Council (NRC) (US). When weather matters, science and service to meet critical societal needs. Washington DC: National Academies Press (US); 2010.

University of Utah, Department of Atmospheric Sciences, Mesowest Region Web page. Salt Lake City, (UT); 2014. [accessed 27 August 2014] http://mesowest.utah.edu/cgi-bin/droman/mesomap.cgi?state=NM&rawsflag=3.

Weather Channel, Monthly Average for White Sands Missile Range, NM (88002) Web page. Atlanta (GA) (US); 2014. [accessed 27 August 2014] http://www.weather.com/weather/wxclimatology/monthly/88002.

## Appendix A. Meteorological Sensor Array (MSA)-Phase I, CR23X Micrologger Program

This appendix appears in its original form, without editorial change.

```
;{CR23X}
;{CR23X} THIS DATALOGGER GOES TO TOWER 4 ON THE EAST SIDE OF THE ARRAY.
;****************************************************************************
;  Program:   MSA Thermo CAL  (4653_MSA.CSI)
;  Last Rev.  2014 MAR 05
;  POC:       Brice
;  CMP3 serial # 092102
;****************************************************************************

*Table 1 Program
  01: 5.0000    Execution Interval (seconds)

;****************************************************************************
;PRESSURE SENSOR (PTB-101B - Vaisala) S/N X1720009

1:  Do (P86)
 1: 45       Set Port 5 High

2:  Excite-Delay (SE) (P4)
 1: 1        Reps
 2: 25       5000 mV, 60 Hz Reject, Fast Range (Delay must be 0)
 3: 7        SE Channel
 4: 3        Excite all reps w/Exchan 3
 5: 100      Delay (0.01 sec units)
 6: 5000     mV Excitation
 7: 1        Loc [ Pressure  ]
 8: .184     Multiplier
 9: 600      Offset

; Pressure transmitter mounts inside the enclosure next to the logger.
;  Blue to SE7
;  Yellow and Clear to Ground
;  Red to 12V
;  Green to C5
;  Black to Control Ground

3:  Do (P86)
 1: 55       Set Port 5 Low

;****************************************************************************
;TEMPERATURE (T107 - Campbell)

4:  Temp (107) (P11)
 1: 1        Reps
 2: 1        SE Channel
 3: 21       Excite all reps w/E1, 60Hz, 10ms delay
 4: 2        Loc [ TempHi    ]
 5: 1.0      Multiplier
 6: 0.0      Offset

; One sensor:
;  Red to SE1
;  Black to EX1
;  Purple to Ground
;  Clear to Excitation Ground
```

```
;*****************************************************************************
;TEMPERATURE/RH (Rotronic HC2S3 T/RH) S/N 61081155
5:  Do (P86)
 1: 49       Turn On Switched 12V

6:  Delay w/Opt Excitation (P22)
 1: 2        Ex Channel
 2: 0        Delay W/Ex (0.01 sec units)
 3: 300      Delay After Ex (0.01 sec units)
 4: 0        mV Excitation

; The preceding instruction provides a delay for sensor warm up before
measurement.

7:  Volt (SE) (P1)
 1: 1        Reps
 2: 14       1000 mV, Fast Range
 3: 4        SE Channel
 4: 3        Loc [ TempVais  ]
 5: .1       Multiplier
 6: -40      Offset

8:  Volt (SE) (P1)
 1: 1        Reps
 2: 14       1000 mV, Fast Range
 3: 3        SE Channel
 4: 4        Loc [ Hum       ]
 5: .1       Multiplier
 6: 0.0      Offset

; The Rotronic HC2S3 sensor.
;  White to SE3
;  Brown to SE4
;  Clear to Ground
;  Gray to PW AG
;  Green to SW12V
;  Yellow to AG

9:  Do (P86)
 1: 59       Turn Off Switched 12V
;*****************************************************************************
; SOLAR RADIATION (CM3 Pyranometer - Kipp and Zonen)
; Serial Number 092102,  C= 10.48 * E-6 V/W/m^2)

10:  Delay w/Opt Excitation (P22)
 1: 3        Ex Channel
 2: 0        Delay W/Ex (0.01 sec units)
 3: 15       Delay After Ex (0.01 sec units)
 4: 0        mV Excitation

11:  Volt (Diff) (P2)
 1: 1        Reps
 2: 20       Auto, 60 Hz Reject, Slow Range (OS>1.06)
 3: 3        DIFF Channel
 4: 5        Loc [ Solar     ]
 5: 95.419   Multiplier
 6: 0.0      Offset
```

```
; The Kipp and Zonen CM3 pyranometer.
;  White to diff channel 3H
;  Black to 3L
;  Jumper from 3L to Ground
;  Clear to Ground

; Set negative values to zero.

12:  If (X<=>F) (P89)
 1: 5        X Loc [ Solar     ]
 2: 4        <
 3: 0        F
 4: 30       Then Do

     13:  Z=F x 10^n (P30)
      1: 0        F
      2: 0        n, Exponent of 10
      3: 5        Z Loc [ Solar     ]

14:  End (P95)

;***************************************************************************
;BATTERY [Setup uses AC Adapter - Input is 120V @ 60Hz]

15:  Batt Voltage (P10)
 1: 6        Loc [ BatVolt   ]

;***************************************************************************
;PANEL TEMPERATURE

16:  Panel Temperature (P17)
 1: 7        Loc [ PanTemp   ]

;***************************************************************************
;PROCESSING FUNCTIONS START HERE

17:  If time is (P92)
 1: 0        Minutes (Seconds --) into a
 2: 1        Interval (same units as above)
 3: 10       Set Output Flag High (Flag 0)

18:  Set Active Storage Area (P80)^19271
 1: 02       Final Storage Area 2
 2: 004      Array ID

19:  Real Time (P77)^11358
 1: 1220     Year,Day,Hour/Minute (midnight = 2400) ;

20:  Resolution (P78)
 1: 1        High Resolution

21:  Average (P71)^3052
 1: 7        Reps
 2: 1        Loc [ Pressure  ]

22:  Serial Out (P96)
```

```
 1: 56    -- Destination Output

23:  Serial Out (P96)
 1: 80       Destination Output

*Table 2 Program
  02: 0.0000    Execution Interval (seconds)

*Table 3 Subroutines

End Program


-Input Locations-
1 Pressure  5 1 1
2 TempHi    9 1 1
3 TempVais  1 1 1
4 Hum       1 0 1
5 Solar     1 1 2
6 BatVolt   1 0 1
7 PanTemp   1 0 1
8 _________ 1 0 0
9 _________ 1 0 0
10 _________ 1 0 0
11 _________ 1 0 0
12 _________ 0 0 0
13 _________ 0 0 0
14 _________ 0 0 0
15 _________ 0 0 0
16 _________ 0 0 0
17 _________ 0 0 0
18 _________ 0 0 0
19 _________ 0 0 0
20 _________ 0 0 0
21 _________ 0 0 0
22 _________ 0 0 0
23 _________ 0 0 0
24 _________ 0 0 0
25 _________ 0 0 0
26 _________ 0 0 0
27 _________ 0 0 0
28 _________ 0 0 0
-Program Security-
0000
0000
0000
-Mode 4-
-Final Storage Area 2-
250000
-CR10X ID-
0
-CR10X Power Up-
3
-CR10X Compile Setting-
3
-CR10X RS-232 Setting-
-1
```

```
-DLD File Labels-
0
-Final Storage Labels-
0,Year_RTM,11358
0,Day_RTM
0,Hour_Minute_RTM
1,Pressure_AVG~1,3052
1,TempHi_AVG~2
1,TempVais_AVG~3
1,Hum_AVG~4
1,Solar_AVG~5
1,BatVolt_AVG~6
1,PanTemp_AVG~7
2,4,19271
```

# Appendix B. Meteorological Sensor Array (MSA)-Phase I, Pre-Data-Merge Logger Program

This appendix appears in its original form, without editorial change.

Code for recording data logger serial output

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <time.h>
#include <errno.h>
#include <unistd.h>
#include <termios.h>
#include <sys/stat.h>
#include <asm/ioctls.h>
#include <linux/serial.h>
#include <signal.h>
#include <sys/time.h>

#define BAUDRATE B19200
#define FILEDIR "/home/daq/data"

#define FALSE 0
#define TRUE  1
#define SIGNALDIR "/home/daq/data/signals/"             // dir for trigger

extern FILE *fopen();
extern int fclose();

int mainloop(FILE*, FILE*, int);
int get_timestamp(char*);
int get_message(FILE*, char*);
int set_ports(char*);
void flush_line(FILE*);
void port_error(char*);
int semaphore(char*, int);
int check_dev(char*);
void set_station(int, char **, int*, int*, int*, char*);
int catoi(char *);

int main(int argc, char* argv[])
{
        int n, rtn, stop, newfile, xx=0, yy=0, sensor_ht=0, TZ=-7, hour, lhr;
        char amline[70], portname[] = "/dev/ttyUSB0", datafilename[50], fileheader[50];
        struct tm *gmt;
        time_t start_time;
        FILE *com_port, *data_file;
        struct stat attributes;

        set_station(argc, argv, &xx, &yy, &sensor_ht, portname);
        printf("%d, %d, %d, %s\n", xx, yy, sensor_ht, portname);

        n = set_ports(portname);
        com_port = fopen(portname,"rw");
        if (com_port == NULL) port_error(portname);
//printf("flushing first line\n");
          flush_line(com_port);

          stop = 0;
          while(!stop)
          {
                    start_time = time(NULL);
                    gmt = gmtime(&start_time);
                    lhr = gmt->tm_hour + TZ;
                    if(lhr < 0) lhr += 24;
                    sprintf(datafilename,"%s/%04d%02d%02d_%02d00_%02d%02d_logger.txt", FILEDIR,
1900+gmt->tm_year, 1+gmt->tm_mon, gmt->tm_mday, gmt->tm_hour, xx, yy);

                    sprintf(fileheader,"%04d%02d%02d %03d %02d00 %02d00 %02d:%02d",1900+gmt-
>tm_year, 1+gmt->tm_mon, gmt->tm_mday, gmt->tm_yday+1, gmt->tm_hour, lhr, xx, yy);

/*                  sprintf(datafilename,"%s/%04d%02d%02d_%02d%02d_%02d%02d_logger.txt",
FILEDIR, 1900+gmt->tm_year, 1+gmt->tm_mon, gmt->tm_mday, gmt->tm_hour, gmt->tm_min, xx,
yy);
                    sprintf(fileheader,"%04d%02d%02d %03d %02d%02d %02d%02d
%02d:%02d",1900+gmt->tm_year, 1+gmt->tm_mon, gmt->tm_mday, gmt->tm_yday+1, gmt->tm_hour,
gmt->tm_min, lhr, gmt->tm_min, xx, yy);*/
//                  printf("wind file = %s\nheader = %s\n",datafilename, fileheader);

                    rtn = stat(datafilename, &attributes);
                    if(rtn == 0)
                    {
                              data_file = fopen(datafilename,"a");
                    }
                    else
                    {
                              data_file = fopen(datafilename,"w");
                              fprintf(data_file,"%s\n", fileheader);
                    }
                    fclose(data_file);
                    newfile = 0;
                    hour = gmt->tm_hour;
//printf("loop start\n");
                    while((!newfile) && (!stop))
                    {
                              data_file = fopen(datafilename,"a");
                              newfile = mainloop(com_port, data_file, hour);
                              fclose(data_file);
                              stop = semaphore("stop_logger", 0);
                              if((!newfile) && (!stop)) sleep(58);
                    }
          }
printf("logger stopping\n");
return(0);
}
```

```
int mainloop(FILE *com_port, FILE *wind_file, int start_hr)
{
int rtn, length=0, cs, checksum, n, hr;
char amline[50], timestamp[40];

        hr = get_timestamp(timestamp);
        if (hr == start_hr)
        {
//printf("ts = %s, hr = %d", timestamp, hr);
                get_message(com_port, amline);
//printf("line = %s\n", amline);
//              printf("%s %s\n", timestamp, amline);
                fprintf(wind_file, "%s %s\n", timestamp, amline);
                return(0);
        }
        else
        {
//              printf("file end\n");
                return(1);
        }
}

void flush_line(FILE *com_port)
{
char line[50];
        get_message(com_port, line);
        sleep(58);
}

int get_message(FILE *port, char *line)
{
        int i, Retry;
        char c;
        i = 0;
        Retry = 0;
//printf("get msg strt:");
        while(!Retry)
        {
                c = '\0';
                c = fgetc(port);
                if (c == '\n')
                {
                        line[i] = '\0';
                        Retry = 1;
                }
                else
                {
                        if (((c > '\037') && (c < '\163')) || (c == 9))
                        {
                                line[i] = c;
                                i++;
```

```
				}
			}
		}
//printf(" get msg rtn\n");
		return(i);
}

int get_timestamp(char *stamp)
{
		int lhr;
		struct tm *gmt;
		time_t get_time;
		struct timeval tv;
		struct timezone tz;

		gettimeofday(&tv, &tz);
		gmt = gmtime(&(tv.tv_sec));
		if (gmt->tm_hour > 6) lhr = gmt->tm_hour - 7;
		else lhr = gmt->tm_hour + 17;
/*		sprintf(stamp,"%02d%02d%02d %02d%02d %02d.%03d",gmt->tm_year-100, 1+gmt->tm_mon,
gmt->tm_mday, gmt->tm_hour, gmt->tm_min, gmt->tm_sec, tv.tv_usec/1000);*/
		sprintf(stamp,"%02d.%03d", gmt->tm_min*60+gmt->tm_sec, tv.tv_usec/1000);

		return(gmt->tm_hour);
}

void set_station(int argc, char **argv, int* x, int* y, int* h, char* portname)
{
int i,com;

		for (i=1; i < argc; i++)
		{
//		  printf("stop 1%s\n",argv[i]);

			if ((*argv[i] == 'X')||(*argv[i] == 'x'))
			{
//				Setting Site x coord
				*x = catoi(argv[++i]);
			}
			if ((*argv[i] == 'Y')||(*argv[i] == 'y'))
			{
//				Setting Site y coord
				*y = catoi(argv[++i]);
			}
			if ((*argv[i] == 'H')||(*argv[i] == 'h'))
			{
//				Setting sensor height
				*h = catoi(argv[++i]);
			}
			if ((*argv[i] == 'P')||(*argv[i] == 'p'))
			{
```

```
//                              Setting serial port
                                com = catoi(argv[++i]);
                                if ((com <= 7) && (com >= 0)) portname[11]+=com;
                        }
        }
}

int set_ports(char* portname)
{
        int term, r1, r2;
        struct termios set_term;

        term = open(portname, O_RDWR | O_NOCTTY | O_NDELAY);
        if (term == -1) return(-1);

        r1 = ioctl(term,TCGETS,&set_term);
        if (r1 < 0) return(-2);

        set_term.c_cflag = BAUDRATE | CS8 | CREAD | CLOCAL;
        set_term.c_iflag = IGNPAR;
        set_term.c_lflag = 0;

        r2 = ioctl(term,TCSETS,&set_term);
        if (r2 < 0) return(-3);
        close(term);  //*/

        return(0);
}

void port_error(char *portname)
{
        printf("%s not available\n",portname);
}

int check_dev(char *portname)
{
struct stat attributes;
int rtn=1; //, n;

        rtn = stat(portname, &attributes);
        if (rtn != 0) return(0);

        return(1);
}

int semaphore(char *signalname, int flag)
{
        int rtn = 0;
        char command[80];
        FILE *flagfile;
```

```
	if(flag)
	{
//		sprintf(command,"echo \"1\" > %s%s\n", SIGNALDIR, signalname);
//		system(command);
		sprintf(command,"%s%s", SIGNALDIR, signalname);
//		printf("%s\n", command);
		flagfile = fopen(command, "w");
		fprintf(flagfile,"1");
		fclose(flagfile);
		rtn = 1;
	}
	else
	{
		sprintf(command,"%s%s", SIGNALDIR, signalname);
		rtn = check_dev(command);
		if(rtn)
		{
//			sprintf(command,"rm -f %s%s\n", SIGNALDIR, signalname);
//			system(command);
			sprintf(command,"%s%s", SIGNALDIR, signalname);
			remove(command);
		}
	}

	return(rtn);
}

int catoi(char *string)
{
	int num, pow, sign, i;

	sign = 1;
	pow = 10;
	num = 0;
	for (i = 0; string[i] != '\0'; i++)
	{
		if (string[i] >= '0' && string[i] <= '9')
		{
			if (i == 0)
				num = string[i] - '0';
			else
				num = num * pow + (string[i] - '0');
		}
		else
		{
			if (string[i] == '-')
				sign = -1;
		}
	}
	return (sign * num);
}
```

INTENTIONALLY LEFT BLANK.

## Appendix C. Meteorological Sensor Array (MSA)-Phase I, Pre-Data-Merge Sonic Program

This appendix appears in its original form, without editorial change.

Code for recording sonic output

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <time.h>
#include <errno.h>
#include <unistd.h>
#include <termios.h>
#include <sys/stat.h>
#include <asm/ioctls.h>
#include <linux/serial.h>
#include <signal.h>
#include <sys/time.h>

#define BAUDRATE B19200
#define FILEDIR "/home/daq/data"
#define SIGNALDIR "/home/daq/data/signals/"            // dir for trigger

#define FALSE 0
#define TRUE  1

extern FILE *fopen();
extern int fclose();

int mainloop(FILE*, FILE*, int);
int get_timestamp(char*);
int get_message(FILE*, char*);
int set_ports(char*);
void flush_line(FILE*);
void port_error(char*);
int semaphore(char*, int);
int check_dev(char*);
void set_station(int, char **, int*, int*, int*, char*);
int catoi(char *);

int main(int argc, char* argv[])
{
        int n, rtn, stop, newfile, xx=0, yy=0, sensor_ht=0, TZ=-7, hour, lhr, offset;
        char amline[70], portname[] = "/dev/ttyUSB0", datafilename[50], fileheader[50], semafile[30],
tst;
        struct tm *gmt;
        time_t start_time;
        FILE *com_port, *data_file;
        struct stat attributes;

        set_station(argc, argv, &xx, &yy, &sensor_ht, portname);
        printf("%d, %d, %d, %s\n", xx, yy, sensor_ht, portname);

        n = set_ports(portname);
```

```
	sprintf(semafile,"stop_sonic_%02d", sensor_ht);
	com_port = fopen(portname,"rw");
	if (com_port == NULL) port_error(portname);
	flush_line(com_port);

	stop = 0;
	while(!stop)
	{
		start_time = time(NULL);
		gmt = gmtime(&start_time);
		lhr = gmt->tm_hour + TZ;
		if(lhr < 0) lhr += 24;
/*		sprintf(datafilename,"%s/%04d%02d%02d_%02d%02d_%02d%02d_%02d_sonic.txt",
FILEDIR, 1900+gmt->tm_year, 1+gmt->tm_mon, gmt->tm_mday, gmt->tm_hour, gmt->tm_min, xx, yy,
sensor_ht);
		sprintf(fileheader,"%04d%02d%02d %03d %02d%02d %02d%02d %02d:%02d
%02d",1900+gmt->tm_year, 1+gmt->tm_mon, gmt->tm_mday, gmt->tm_yday+1, gmt->tm_hour, gmt-
>tm_min, lhr, gmt->tm_min, xx, yy, sensor_ht);*/
		sprintf(datafilename,"%s/%04d%02d%02d_%02d00_%02d%02d_%02d_sonic.txt",
FILEDIR, 1900+gmt->tm_year, 1+gmt->tm_mon, gmt->tm_mday, gmt->tm_hour, xx, yy, sensor_ht);
		sprintf(fileheader,"%04d%02d%02d %03d %02d00 %02d00 %02d:%02d
%02d",1900+gmt->tm_year, 1+gmt->tm_mon, gmt->tm_mday, gmt->tm_yday+1, gmt->tm_hour, lhr,
xx, yy, sensor_ht);
		printf("wind file = %s\nheader = %s\n",datafilename, fileheader);

		rtn = stat(datafilename, &attributes);
		if(rtn == 0)
		{
			data_file = fopen(datafilename,"r+");
			fseek(data_file, -1, SEEK_END);
			offset = 0;
			while(fgetc(data_file) != 10)
			{
				fseek(data_file, -2, SEEK_CUR);
				offset--;
			}
			fseek(data_file, offset, SEEK_END);
		}
		else
		{
			data_file = fopen(datafilename,"w");
			fprintf(data_file,"%s\n", fileheader);
		}
		newfile = 0;
		hour = gmt->tm_hour;
		while((!newfile) && (!stop))
		{
			newfile = mainloop(com_port, data_file, hour);
			stop = semaphore(semafile, 0);
		}
		fclose(data_file);
		}
printf("sonic stopping\n");
return(0);
}

int mainloop(FILE *com_port, FILE *wind_file, int start_hr)
{
int rtn, length=0, cs, checksum, n, hr, stop;
char amline[50], timestamp[40];

		hr = get_timestamp(timestamp);
		if (hr == start_hr)
		{
//printf("ts = %s, hr = %d", timestamp, hr);
				get_message(com_port, amline);
//printf("%s\n", amline);
				fprintf(wind_file, "%s %s\n", timestamp, amline);
				return(0);
		}
		else
		{
//				printf("file end\n");
				return(1);
		}
}

void flush_line(FILE *com_port)
{
char line[50];
		get_message(com_port, line);
}

int get_message(FILE *port, char *line)
{
		int i, Retry;
		char c;

		i = 0;
		Retry = 0;
		while(!Retry)
		{
				c = '\0';
				c = fgetc(port);
				if ((c == '\r')||(c == '\n'))
				{
						line[i] = '\0';
						Retry = 1;
				}
				else
				{
						if (((c > '\037') && (c < '\163')) || (c == 9))
```

```
                    {
                            line[i] = c;
                            i++;
                    }
            }
    }
    return(i);
}

int get_timestamp(char *stamp)
{
    int lhr;
    struct tm *gmt;
    time_t get_time;
    struct timeval tv;
    struct timezone tz;

    gettimeofday(&tv, &tz);
    gmt = gmtime(&(tv.tv_sec));
    if (gmt->tm_hour > 6) lhr = gmt->tm_hour - 7;
    else lhr = gmt->tm_hour + 17;
/*  sprintf(stamp,"%02d%02d%02d %02d%02d %02d.%03d",gmt->tm_year-100, 1+gmt->tm_mon,
gmt->tm_mday, gmt->tm_hour, gmt->tm_min, gmt->tm_sec, tv.tv_usec/1000);*/
    sprintf(stamp,"%02d.%03d", gmt->tm_min*60+gmt->tm_sec, tv.tv_usec/1000);

    return(gmt->tm_hour);
}

void set_station(int argc, char **argv, int* x, int* y, int* h, char* portname)
{
int i,com;

    for (i=1; i < argc; i++)
    {
            if ((*argv[i] == 'X')||(*argv[i] == 'x'))
            {
//                  Setting Site x coord
                    *x = catoi(argv[++i]);
            }
            if ((*argv[i] == 'Y')||(*argv[i] == 'y'))
            {
//                  Setting Site y coord
                    *y = catoi(argv[++i]);
            }
            if ((*argv[i] == 'H')||(*argv[i] == 'h'))
            {
//                  Setting sensor height
                    *h = catoi(argv[++i]);
            }
            if ((*argv[i] == 'P')||(*argv[i] == 'p'))
            {
```

```
//                              Setting serial port
                                com = catoi(argv[++i]);
                                if ((com <= 7) && (com >= 0)) portname[11]+=com;
                        }
        }
}

int set_ports(char* portname)
{
        int term, r1, r2;
        struct termios set_term;

        term = open(portname, O_RDWR | O_NOCTTY | O_NDELAY);
        if (term == -1) return(-1);

        r1 = ioctl(term,TCGETS,&set_term);
        if (r1 < 0) return(-2);

        set_term.c_cflag = BAUDRATE | CS8 | CREAD | CLOCAL;
        set_term.c_iflag = IGNPAR;
        set_term.c_lflag = 0;

        r2 = ioctl(term,TCSETS,&set_term);
        if (r2 < 0) return(-3);
        close(term);  //*/

        return(0);
}

void port_error(char *portname)
{
        printf("%s not available\n",portname);
}

int check_dev(char *portname)
{
struct stat attributes;
int rtn=1; //, n;

        rtn = stat(portname, &attributes);
        if (rtn != 0) return(0);

        return(1);
}

int semaphore(char *signalname, int flag)
{
        int rtn = 0;
        char command[80];
        FILE *flagfile;
```

```
	if(flag)
	{
//		sprintf(command,"echo \"1\" > %s%s\n", SIGNALDIR, signalname);
//		system(command);
		sprintf(command,"%s%s", SIGNALDIR, signalname);
		printf("%s\n", command);
		flagfile = fopen(command, "w");
		fprintf(flagfile,"1");
		fclose(flagfile);
		rtn = 1;
	}
	else
	{
		sprintf(command,"%s%s", SIGNALDIR, signalname);
		rtn = check_dev(command);
		if(rtn)
		{
//			sprintf(command,"rm -f %s%s\n", SIGNALDIR, signalname);
//			system(command);
			sprintf(command,"%s%s", SIGNALDIR, signalname);
			remove(command);
		}
	}
//printf("semaphore returns %d", rtn);
	return(rtn);
}

int catoi(char *string)
{
	int num, pow, sign, i;

	sign = 1;
	pow = 10;
	num = 0;
	for (i = 0; string[i] != '\0'; i++)
	{
		if (string[i] >= '0' && string[i] <= '9')
		{
			if (i == 0)
				num = string[i] - '0';
			else
				num = num * pow + (string[i] - '0');
		}
		else
		{
			if (string[i] == '-')
				sign = -1;
		}
	}
	return (sign * num);
}
```

INTENTIONALLY LEFT BLANK.

## Appendix D. Meteorological Sensor Array (MSA)-Phase I, Pre-Data-Merge Averaging Program

This appendix appears in its original form, without editorial change.

Code for 1 minute averages

```
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <fcntl.h>
#include <time.h>
#include <errno.h>
#include <unistd.h>
#include <termios.h>
#include <sys/stat.h>
#include <asm/ioctls.h>
#include <linux/serial.h>
#include <signal.h>
#include <sys/time.h>

#define BAUDRATE B19200
#define FILEDIR "/home/daq/data"

#define FALSE 0
#define TRUE  1
#define SIGNALDIR "/home/daq/data/signals/"              // dir for trigger
#define LOGFILE "/home/daq/data/logs/datalog"                       // data management log file
#define INFOFILE "/home/daq/data/logs/info"              // data management log file

extern FILE *fopen();
extern int fclose();

int get_timestamp(char*);
int process_all(long, int, int, int, char*);
int process_logger(char*);
int process_sonic(int, char*);
int merge_process(char*);
int semaphore(char*, int);
int check_dev(char*);

int main(int argc, char* argv[])
{
        int n, stop, newfile, xx=5, yy=5, sensor_ht=0, TZ=-7, hour, lhr, day, month, filetime;
        char datafilemove[120], datafilecheck[120], mknewdir[80], date[25], ts[30], newdir[70],
fileroot[50];
        long startdate;
        struct tm *gmt;
        time_t start_time;
        FILE *logfile, *infofile;
        struct stat attributes;

        start_time = time(NULL);
        gmt = gmtime(&start_time);
        lhr = gmt->tm_hour + TZ;
        hour = gmt->tm_hour-1;
```

```
		day = gmt->tm_mday;
		month = 1+gmt->tm_mon;
		if(hour < 0)
		{
			hour += 24;
			day -= 1;
			if (day == 0)
			{
				month -= 1;
				if(month == 4) day == 30;
				else day = 31;
			}
		}
		startdate = (1900+gmt->tm_year)*10000+month*100+day;
		filetime = hour*100;
		sprintf(fileroot, "/home/daq/data/");

		get_timestamp(ts);
		sprintf(date, "%04d%02d%02d", 1900+gmt->tm_year, month, day);
		sprintf(newdir,"%s/%s", FILEDIR, date);
		sprintf(mknewdir,"mkdir %s", newdir);

		sprintf(datafilemove,"mv %s_%02d* %s", newdir, hour, newdir);

		if(check_dev(newdir) == 0) system(mknewdir);
		system(datafilemove);

//		sprintf(datafilemove,"ls %s_%02d00_* > ", newdir, hour, newdir);
		infofile = fopen(INFOFILE,"r");
		fscanf(infofile, "%02d%02d", &xx, &yy);
		fclose(infofile);

		process_all(startdate, filetime, xx, yy, fileroot);
		logfile = fopen(LOGFILE,"a");
		fprintf(logfile,"%s: %s moved\n", ts, date);
		fclose(logfile);

return(0);
}

int get_timestamp(char *stamp)
{
		int lhr;
		struct tm *gmt;
		time_t get_time;
		struct timeval tv;
		struct timezone tz;

		gettimeofday(&tv, &tz);
		gmt = gmtime(&(tv.tv_sec));
		if (gmt->tm_hour > 6) lhr = gmt->tm_hour - 7;
```

```
	else lhr = gmt->tm_hour + 17;
	sprintf(stamp,"%02d%02d%02d %02d%02d %02d.%03d",gmt->tm_year-100, 1+gmt->tm_mon,
gmt->tm_mday, gmt->tm_hour, gmt->tm_min, gmt->tm_sec, tv.tv_usec/1000);
//	sprintf(stamp,"%02d.%03d", gmt->tm_min*60+gmt->tm_sec, tv.tv_usec/1000);

	return(gmt->tm_hour);
}


int process_all(long startdate, int filetime, int xx, int yy, char *fileroot)
{
int rtn, level, missing=0;
char filebase[60], sonicfile[80], loggerfile[80];
struct stat attributes;

	sprintf(filebase,"%s%08d/%08d_%04d_%02d%02d", fileroot, startdate, startdate, filetime, xx,
yy);


	for (level=0;level<2;level++)
	{
		sprintf(sonicfile,"%s_%02d_sonic.txt", filebase, level*8+2);
		rtn = stat(sonicfile, &attributes);
		if(!rtn) process_sonic(level*8+2, filebase);
		else
		{
			printf("%s missing\n", sonicfile);
			missing++;
		}
	}

	sprintf(loggerfile,"%s_logger.txt", filebase);
	rtn =  stat(loggerfile, &attributes);
	if(!rtn) process_logger(filebase);
	else
	{
		printf("%s missing\n", loggerfile);
		missing++;
	}


	if(!missing) merge_process(filebase);

	return(1);
}

int merge_process(char *filebase)
{
FILE *logger, *sonic2, *sonic10, *output;
int err2, err10, ftime, day, min, ht2, xx, yy, alt=0;
float  press, rh, volt, panel_temp, log_second, t2, t10, solar;
float ws2, wd2, ws10, wd10, u2, v2, w2, u10, v10, w10, ts2, ts10, c_2, c_10, hr2, hr10, lhr2, lhr10, hr0,
lhr0;
double lat=0, lon=0;
char logfile[50], mergefile[80], sonicfile2[80], sonicfile10[80], tower[9];
long date;
//time_t utime;
//struct tm *ltime;

                sprintf(sonicfile2,"%s_02_sonic.avg.txt", filebase);
                sprintf(sonicfile10,"%s_10_sonic.avg.txt", filebase);
//              sprintf(logfile,"%s_logger.txt", filebase);
                sprintf(logfile,"%s_logger.avg.txt", filebase);
                sprintf(mergefile,"%s_merged.txt", filebase);
                logger=fopen(logfile,"r");
                sonic2=fopen(sonicfile2,"r");
                sonic10=fopen(sonicfile10,"r");
                output=fopen(mergefile,"w");
                fscanf(logger,"%*d%*d%*d%*d%2d:%2d", &xx, &yy);
                if(xx == 3)
                {
                        lat = 32.3958;
                        lon = -106.4729;
                        alt = 1286;
                }
                if(xx == 2)
                {
                        lat = 32.3959;
                        lon = -106.4740;
                        alt = 1287;
                }
                if(xx == 1)
                {
                        if(yy == 1)
                        {
                                lat = 32.3949;
                                lon = -106.4825;
                                alt = 1306;
                        }
                        if(yy == 2)
                        {
                                lat = 32.3957;
                                lon = -106.4825;
                                alt = 1308;
                        }
                        if(yy == 3)
                        {
                                lat = 32.3966;
                                lon = -106.4825;
                                alt = 1307;
                        }
//                      if(yy == 4) lat = 99.9999;
```

```
                    }
                    fscanf(sonic2,"%ld%d%d%*d%s%d", &date, &day, &ftime, tower, &ht2);
                    fscanf(sonic10,"%*d%*d%*d%*d%*s%*d");
                    fprintf(output,"%08d %03d %04d -7 %02d:%02d %7.4f %9.4f %d\n", date, day, ftime,
xx, yy, lat, lon, alt);
                    for (min=0;min<60;min++)
                    {
//                              fscanf(logger,"%f%*d,%*d,%*d,%*d,%f,%f,%f,%f,%f,%f,%f", &log_second,
&press, &t10, &t2, &rh, &solar, &volt, &panel_temp);
                              fscanf(logger,"%f%f%f%f%f%f%f%f", &log_second, &press, &t10, &t2, &rh,
&solar, &volt, &panel_temp);
                              fscanf(sonic2,"%f%f%f%f%f%f%f%f%f%d", &hr2, &lhr2, &ws2, &wd2, &u2,
&v2, &w2, &ts2, &c_2, &err2);
                              fscanf(sonic10,"%f%f%f%f%f%f%f%f%f%d", &hr10, &lhr10, &ws10, &wd10,
&u10, &v10, &w10, &ts10, &c_10, &err10);
                              hr0 = hr2;
                              lhr0 = lhr2;
                              fprintf(output,"%6.3f %6.3f %6.3f %7.3f %8.3f %6.3f %5.2f %5.1f %5.2f %5.2f
%5.2f %5.2f %6.2f %4d ", hr0, lhr0,
                                        press, rh, solar, t2, ws2, wd2, u2, v2, w2, ts2, c_2, err2);
                              fprintf(output,"%6.3f %5.2f %5.1f %5.2f %5.2f %5.2f %5.2f %6.2f %4d %6.3f
%6.3f\n", t10, ws10, wd10, u10, v10, w10, ts10, c_10, err10, volt, panel_temp);
                    }
                    fclose(sonic2);
                    fclose(sonic10);
                    fclose(logger);
                    fclose(output);
          return(1);
}

int process_sonic(int hh, char *filebase)
{
FILE *input, *output;
int sample, errsum, min, min_start, rtn, lhr, uhr, height, day, invalid;
float tstamp;
long date;
double t_sum, u_sum, v_sum, w_sum, c_sum, u, v, w, temp, c, u_avg, v_avg, w_avg, t_avg, c_avg,
ws_avg, wd_avg, sd_sum=0;
char input_filename[70], output_filename[70], err[10], tower[6];

          sprintf(input_filename,"%s_%02d_sonic.txt",filebase, hh);
          sprintf(output_filename,"%s_%02d_sonic.avg.txt",filebase, hh);
          input=fopen(input_filename,"r");
          output=fopen(output_filename,"w");
          fscanf(input,"%ld%d%d%d%s%d", &date, &day, &uhr, &lhr, tower, &height);
//        day = 71; // temp date fix
//        fscanf(input,"%ld%d%d%s%d", &date, &uhr, &lhr, tower, &height);
          fprintf(output,"%08d %03d %04d %04d %s %02d\n", date, day, uhr, lhr, tower, height);
          uhr/=100;
          lhr/=100;
          min_start = 0;
```

```
	min = 0;
	u_sum = 0;
	v_sum = 0;
	w_sum = 0;
	t_sum = 0;
	c_sum = 0;
	errsum = 0;
	sample = 0;

	rtn = fscanf(input,"%f%lf%lf%lf%lf%lf%s", &tstamp, &u, &v, &w, &temp, &c, err);
	while(rtn != EOF)
	{
		min = (int)tstamp/60;
		if(min > min_start)
		{
			if(sample > 0)
			{
				u_avg = -1. * u_sum / sample;
				v_avg = -1. * v_sum / sample;
				w_avg = w_sum / sample;
				t_avg = t_sum / sample;
				c_avg = c_sum / sample;
				u_sum = 0;
				v_sum = 0;
				w_sum = 0;
				t_sum = 0;
				c_sum = 0;
				ws_avg = pow((u_avg*u_avg)+(v_avg*v_avg), 0.5);
				if(v_avg == 0)
				{
					if(u_avg > 0) wd_avg = 270;
					else wd_avg = 90;
				}
				else
				{
					wd_avg = atan(u_avg/v_avg)*180/3.1416;
					if(v_avg > 0) wd_avg += 180;
					else if(u_avg > 0) wd_avg += 360;
				}
				fprintf(output, "%6.3f %6.3f %6.2f %5.1f %6.2f %6.2f %6.2f %6.2f
%6.2f %4d\n", (float)uhr+(float)min_start/60, (float)lhr+(float)min_start/60,
					ws_avg, wd_avg, u_avg, v_avg, w_avg, t_avg, c_avg, errsum);
			}
			else fprintf(output, "%6.3f %6.3f -99.99 -99.9 -99.99 -99.99 -99.99 -99.99 -99.99
%4d\n", (float)uhr+(float)min_start/60, (float)lhr+(float)min_start/60, errsum);

			if(min > min_start+1)
			{
				while(min > ++min_start) fprintf(output, "%6.3f %6.3f -99.99 -99.9 -
99.99 -99.99 -99.99 -99.99 -99.99 %4d\n",
					(float)uhr+(float)min_start/60, (float)lhr+(float)min_start/60, 0);
				}

				sample = 0;
				errsum = 0;
				min_start = min;
			}
			invalid = strcmp(err, "0");
			if(invalid) errsum++;
			else
			{
				sample++;
				u_sum += u;
				v_sum += v;
				w_sum += w;
				t_sum += temp;
				c_sum += c;
			}
			rtn = fscanf(input,"%f%lf%lf%lf%lf%lf%s", &tstamp, &u, &v, &w, &temp, &c, err);
		}

		if(sample > 0)
		{
			u_avg = -1. * u_sum / sample;
			v_avg = -1. * v_sum / sample;
			w_avg = w_sum / sample;
			t_avg = t_sum / sample;
			c_avg = c_sum / sample;
			u_sum = 0;
			v_sum = 0;
			w_sum = 0;
			t_sum = 0;
			c_sum = 0;
			ws_avg = pow((u_avg*u_avg)+(v_avg*v_avg), 0.5);
			if(v_avg == 0)
			{
				if(u_avg > 0) wd_avg = 270;
				else wd_avg = 90;
			}
			else
			{
				wd_avg = atan(u_avg/v_avg)*180/3.1416;
				if(v_avg > 0) wd_avg += 180;
				else if(u_avg > 0) wd_avg += 360;
			}
			fprintf(output, "%6.3f %6.3f %6.2f %5.1f %6.2f %6.2f %6.2f %6.2f %6.2f %4d\n",
	(float)uhr+(float)min_start/60, (float)lhr+(float)min_start/60,
				ws_avg, wd_avg, u_avg, v_avg, w_avg, t_avg, c_avg, errsum);
		}
		else if(min_start < 60) fprintf(output, "%6.3f %6.3f -99.99 -99.9 -99.99 -99.99 -99.99 -99.99 -
	99.99 %4d\n",
			(float)uhr+(float)min_start/60, (float)lhr+(float)min_start/60, errsum);
	while(++min_start < 60) fprintf(output, "%6.3f %6.3f -99.99 -99.9 -99.99 -99.99 -99.99 -99.99 -
99.99 %4d\n",
		(float)uhr+(float)min_start/60, (float)lhr+(float)min_start/60, 0);

	fclose(input);
	fclose(output);

	return(1);
}

int process_logger(char *filebase)
{
FILE *input, *output;
int sample, min, min_start, rtn, lhr, uhr, day;
float  press, rh, volt, panel_temp, log_second, t2, t10, solar;
float  press_sum, rh_sum, volt_sum, panel_temp_sum, t2_sum, t10_sum, solar_sum;
float  press_avg, rh_avg, volt_avg, panel_temp_avg, t2_avg, t10_avg, solar_avg;
long date;
char input_filename[80], output_filename[80], tower[6];

	sprintf(input_filename,"%s_logger.txt",filebase);
	sprintf(output_filename,"%s_logger.avg.txt",filebase);
	input=fopen(input_filename,"r");
	output=fopen(output_filename,"w");
	fscanf(input,"%ld%d%d%d%s", &date, &day, &uhr, &lhr, tower);
	fprintf(output,"%08d %03d %04d %04d %s\n", date, day, uhr, lhr, tower);
	uhr/=100;
	lhr/=100;
	min_start = 0;
	min = 0;
	press_sum = 0;
	t10_sum = 0;
	t2_sum = 0;
	rh_sum = 0;
	solar_sum = 0;
	volt_sum = 0;
	panel_temp_sum = 0;
	sample=0;

	rtn = fscanf(input,"%f%*d,%*d,%*d,%*d,%f,%f,%f,%f,%f,%f,%f", &log_second, &press, &t10,
&t2, &rh, &solar, &volt, &panel_temp);
	while(rtn != EOF)
	{
		min = (int)log_second/60;
		if(min > min_start)
		{
			if(sample > 0)
			{
				press_avg = press_sum / sample;
				t10_avg = t10_sum / sample;
```

```
                                        t2_avg = t2_sum / sample;
                                        rh_avg = rh_sum / sample;
                                        solar_avg = solar_sum / sample;
                                        volt_avg = volt_sum / sample;
                                        panel_temp_avg = panel_temp_sum / sample;
                                        press_sum = 0;
                                        t10_sum = 0;
                                        t2_sum = 0;
                                        rh_sum = 0;
                                        solar_sum = 0;
                                        volt_sum = 0;
                                        panel_temp_sum = 0;

                                        if(press_avg == 0.) press_avg = (float)-99.999;

                                        fprintf(output, "%6.3f %7.3f %7.3f %7.3f %7.3f %8.3f %6.3f %6.3f\n",
(float)uhr+(float)min_start/60,
                                                press_avg, t10_avg, t2_avg, rh_avg, solar_avg, volt_avg,
panel_temp_avg);
                                }
                                else fprintf(output, "%6.3f -99.999 -99.999 -99.999 -99.999 -999.999 -9.999 -
9.999\n", (float)uhr+(float)min_start/60);

                                if(min > min_start+1)
                                {
                                        while(min > ++min_start) fprintf(output, "%6.3f -99.999 -99.999 -99.999
-99.999 -999.999 -9.999 -9.999\n",
                                                (float)uhr+(float)min_start/60);
                                }

                                sample = 0;
                                min_start = min;
                        }

                        press_sum += press;
                        t10_sum += t10;
                        t2_sum += t2;
                        rh_sum += rh;
                        solar_sum += solar;
                        volt_sum += volt;
                        panel_temp_sum += panel_temp;
                        sample++;

                        rtn = fscanf(input,"%f%*d,%*d,%*d,%*d,%f,%f,%f,%f,%f,%f,%f", &log_second,
&press, &t10, &t2, &rh, &solar, &volt, &panel_temp);
                }

                if(sample > 0)
                {
                        press_avg = press_sum / sample;
                        t10_avg = t10_sum / sample;
```

```
			t2_avg = t2_sum / sample;
			rh_avg = rh_sum / sample;
			solar_avg = solar_sum / sample;
			volt_avg = volt_sum / sample;
			panel_temp_avg = panel_temp_sum / sample;
			press_sum = 0;
			t10_sum = 0;
			t2_sum = 0;
			rh_sum = 0;
			solar_sum = 0;
			volt_sum = 0;
			panel_temp_sum = 0;
			if(press_avg == 0.) press_avg = (float)-99.999;

			fprintf(output, "%6.3f %7.3f %7.3f %7.3f %7.3f %8.3f %6.3f %6.3f\n",
(float)uhr+(float)min_start/60,
				press_avg, t10_avg, t2_avg, rh_avg, solar_avg, volt_avg, panel_temp_avg);
		}
		else if(min_start < 60) fprintf(output, "%6.3f -99.999 -99.999 -99.999 -99.999 -999.999 -9.999 -
9.999\n", (float)uhr+(float)min_start/60);

		while(++min_start < 60) fprintf(output, "%6.3f -99.999 -99.999 -99.999 -99.999 -999.999 -9.999
-9.999\n", (float)uhr+(float)min_start/60);

		fclose(input);
		fclose(output);

		return(1);
}

int check_dev(char *portname)
{
struct stat attributes;
int rtn=1; //, n;

		rtn = stat(portname, &attributes);
		if (rtn != 0) return(0);

		return(1);

}


/*
int semaphore(char *signalname, int flag)
{
		int rtn = 0;
		char command[80];
		FILE *flagfile;

		if(flag)
```

```
	{
//		sprintf(command,"echo \"1\" > %s%s\n", SIGNALDIR, signalname);
//		system(command);
		sprintf(command,"%s%s", SIGNALDIR, signalname);
		printf("%s\n", command);
		flagfile = fopen(command, "w");
		fprintf(flagfile,"1");
		fclose(flagfile);
		rtn = 1;
	}
	else
	{
		sprintf(command,"%s%s", SIGNALDIR, signalname);
		rtn = check_dev(command);
		if(rtn)
		{
//			sprintf(command,"rm -f %s%s\n", SIGNALDIR, signalname);
//			system(command);
			sprintf(command,"%s%s", SIGNALDIR, signalname);
			remove(command);
		}
	}

	return(rtn);
}*/
```

## Appendix E. Data Management Program – Meteorological Sensor Array (MSA)_Mountain Standard Time (MST)_Automate.vbs

This appendix appears in its original form, without editorial change.

The *MSA_MST_Automate.vbs* program calls *MSA_QC_MST.gle* to create midnight to midnight MST time data displays for first level MSA tower data quality control. This program also takes the 1-h MSA Tower Data files over a 24-h (or more) period and assembles them into single 24-h (or more) file in UTC. The program also archives to disk the jpg image of the plots page for each tower. The program does this process for 2 consecutive days creating a page of plots for each tower, for each day.

```
' PROGRAM:  MSA_MST_Automate.vbs
' AUTHOR:   O'Brien / Harrison
' Last Rev: 140605, sh
'
' PURPOSE:  This program calls  MSA_QC_MST.gle to create midnight to midnight MST time data displays for Quality Control of
'MSA Tower Data.  This program 'also takes the MSA Tower Data of one hour files over 24 or more hours  and assembles them into 24
'hour file(s) for UTC.  It also archives to disk the jpg image of the 'plots page for each tower.  The program does this process for two
'consecutive days creating a page of plots for each tower for each day.


Dim archiveMSTFileString
Dim gleFileString
Dim sourceFileString
Dim lastFileString
Dim FSO
Dim sourceFile
Dim archiveMSTFile
Dim gleFile
Dim currentLineNumber

Dim yearStringFirst
Dim monStringFirst
Dim dayStringFirst
Dim utcStringFirst
Dim mstStringFirst
Dim yearStringLast
Dim monStringLast
Dim dayStringLast
Dim utcStringLast
Dim mstStringLast
Dim utcString
Dim towerString

Dim GLE_QC_UTCScript
Dim GLE_QC_MSTScript
Dim GLE_ProgramPath
Dim GLEScriptRunStringUTC
Dim GLEScriptRunStringMST
Dim GLE_Exec_UTC
Dim GLE_Exec_MST

Const ForReading = 1, ForWriting = 2, ForAppending = 8

set FSO = CreateObject("Scripting.FileSystemObject")
set WshShell = CreateObject("WScript.Shell")


rootPath = "C:\MSA\MSA_POST-PoC_Exercise\"
'archivePath = "MSA_Archive\MSA_DateMerged\"
dataPath = "MSA_Data\"
applicationsPath = "MSA_Applications\"
plotsPath = "MSA_Plots\"
LDriveRootPath = "L:\MSA_POST-PoC_Exercise\"
gleFileStringMaster = rootPath & dataPath & "MSA_24hr_GLE_Data.txt"


YMD = InputBox("Enter YEAR/MONTH/DAY of Data Files in Format (YYMMDD):", "MST")
If YMD="" Then
```

```
		WScript.Quit
End If
YMDFirst = YMD

yearString = Left(YMD, 2)
monString = Mid(YMD, 3, 2)
dayString = Mid(YMD, 5, 2)
numTowers = 5
towerString = Array("0101", "0102", "0202", "0302", "0103")
utcString = "07"
yearStringFirst = yearString
monStringFirst = monString
dayStringFirst = dayString


'*** copy LDrive directory of data YMD to working area data dir ***
LDriveDataDirString = LDriveRootPath & dataPath & YMD & "merged"
FSO.CopyFolder LDriveDataDirString, rootPath & dataPath, True

'*** Need to get Julian Day from data file so can calculate YMD of next day to be able to copy directory of data of next day ***
For i = 0 To numTowers-1
		LDriveDataFileString = LDriveDataDirString & "\20" & yearString & monString & dayString & "_" & utcString & "00" & "_" &
towerString(i) & "_merged.txt"
		If FSO.FileExists(LDriveDataFileString) Then
				set LDriveDataFile = FSO.OpenTextFile(LDriveDataFileString, ForReading, True)
				'MsgBox("LDriveDataFile:  " & LDriveDataFileString)
				i = numTowers-1
		Else
				'MsgBox("WARNING: L Drive Data File Not Found" & LDriveDataFileString)
				'WScript.Quit
		End If
Next

FirstLine = LDriveDataFile.ReadLine
julianDayString = Mid(FirstLine, 10, 3)
LDriveDataFile.close


julianDayString = CStr(CInt(julianDayString)+1)
return = YMDFromJulian(yearString, monString, dayString, Int(julianDayString))
YMD = yearString & monString & dayString

'*** copy LDrive directory of data of next day to working area data dir ***
'MsgBox("Next Day  " & YMD)
LDriveDataDirString = LDriveRootPath & dataPath & YMD & "merged"
FSO.CopyFolder LDriveDataDirString, rootPath & dataPath, True



gleFileOpen = 0

For k = 0 To 1		'*** For two days of generating plots - one full 24hr day and next day partial ***

  For i = 0 To numTowers-1

    utcString = "07"
    dayString = dayStringFirst
    monString = monStringFirst
    yearString = yearStringFirst
    YMD = YMDFirst
    TowerDataFiles = False

    For j = 0 To 23

		'*** Open hourly sensor merged data files ***
		Continue = True
		sourceFileString = rootPath & dataPath & YMD & "merged\20" & yearString & monString & dayString & "_" & utcString & "00" &
"_" & towerString(i) & "_merged.txt"
		gleFileString = rootPath & "MSA_Data\MSA_24hr_GLE_Data_" & towerString(i) & ".txt"
		If FSO.FileExists(sourceFileString) Then
```

```
                        set sourceFile = FSO.OpenTextFile(sourceFileString, ForReading, True)
                        'MsgBox("Data File:  " & sourceFileString)
                        TowerDataFiles = True
            Else
                        'MsgBox("WARNING: File Not Found" & sourceFileString)
                        'WScript.Quit
                        Continue = False
            End If


            '*** Merge all hourly files into one ***
            If Continue = True Then
                        If gleFileOpen = 0 Then
                                    set gleFile = FSO.OpenTextFile(gleFileString, ForWriting, True)
                                    gleFileOpen = 1
                        Else
                                    gleFile.close
                                    set gleFile = FSO.OpenTextFile(gleFileString, ForAppending, True)
                                    gleFileOpen = 1
                        End If

                        currentLineNumber = 0

                        Do While Not(sourceFile.AtEndofStream)
                                    currentLineNumber = currentLineNumber + 1
                                    currentLine = sourceFile.ReadLine
                                    If currentLineNumber = 1 Then
                                                julianDayString = Mid(currentLine, 10, 3)
                                                gleFile.WriteLine("!" & currentLine)
                                    Else
                                                gleFile.WriteLine(currentLine)
                                    End If
                        Loop
                        sourceFile.close

            End If  '*** Continue ***

            '*** Set variables so we have the correct file name and directory of next hourly file to read ***
            utcString = CStr(CInt(utcString)+1)
            utcString = string(2 - Len(utcString), "0") & utcString
            If utcString = "24" Then
                        utcString = "00"

                        If TowerDataFiles = True Then    '*** If no data files to read julian day from, then use julian day from previous tower file
which has already been incremented to next day ***
                                    julianDayString = CStr(CInt(julianDayString)+1)
                        End If

                        return = YMDFromJulian(yearString, monString, dayString, Int(julianDayString))
                        YMD = yearString & monString & dayString
                        'MsgBox("Next day: " & yearString & monString & dayString)
            End If

   Next    '***For Loop Hour (j) ***


   If TowerDataFiles = True Then

            '*** Set dir and file name for archiving ***
            'If Not FSO.FolderExists(rootPath & archivePath & YMDFirst) Then
            '           FSO.CreateFolder(rootPath & archivePath & YMDFirst)
            'End If
            'archiveMSTFileString = rootPath & archivePath & YMDFirst & "\MSA_MST_" & yearStringFirst & monStringFirst &
dayStringFirst & "_" & TowerString(i) & ".txt"
            'FSO.CopyFile gleFileString, archiveMSTFileString, True
            FSO.CopyFile gleFileString, gleFileStringMaster, True

            '*** Set up and call GLE script also archive jpg image of plot ***
            GLE_QC_MSTScript = rootPath & applicationsPath & "MSA_QC_MST.gle"
            GLE_ProgramPath = "C:\LRx\Programs\GLE4\bin\gle "
```

```
		'QGLE_ProgramPath = "C:\LRx\Programs\GLE4\bin\qgle "
		'QGLEScriptRunStringMST = QGLE_ProgramPath & GLE_QC_MSTScript
		'Set QGLE_Exec_MST = WshShell.Exec(QGLEScriptRunStringMST)

		GLEScriptRunStringMST = GLE_ProgramPath & "-d jpg " & GLE_QC_MSTScript
		Set GLE_Exec_MST = WshShell.Exec(GLEScriptRunStringMST)

		GLEScriptRunStringMST = GLE_ProgramPath & "/preview " & GLE_QC_MSTScript
		Set GLE_Exec_MST = WshShell.Exec(GLEScriptRunStringMST)

		jpgFileString = rootPath & applicationsPath & "MSA_QC_MST.jpg"
		jpgFileArchiveString = rootPath & plotsPath & YMDFirst & "\MSA_Plots_MST_" & yearStringFirst & monStringFirst & 
dayStringFirst & "_" & TowerString(i) & ".jpg"


		Do While (Not FSO.FileExists(jpgFileString))
			Wscript.sleep(10000)
		Loop

		If FSO.FileExists(jpgFileString) Then
			If Not FSO.FolderExists(rootPath & plotsPath & YMDFirst) Then
				FSO.CreateFolder(rootPath & plotsPath & YMDFirst)
			End If
			Wscript.Echo(jpgFileArchiveString)
			FSO.CopyFile jpgFileString, jpgFileArchiveString, True
			FSO.DeleteFile jpgFileString, True

			' *** Delete No data found txt file from previous day if exists, ie Tower has changed states to working ***
			plotsNoDataFileString = rootPath & plotsPath & YMDFirst & "\MSA_Plots_MST_" & yearStringFirst & monStringFirst
& dayStringFirst & "_" & TowerString(i) & ".txt"
			If FSO.FileExists(plotsNoDataFileString) Then
				FSO.DeleteFile plotsNoDataFileString, True
				'MsgBox("plotsNoDataFileString to delete=" & plotsNoDataFileString)
			End If

			LDrivePlotsFileString = LDriveRootPath & plotsPath & YMDFirst & "\MSA_Plots_MST_" & yearStringFirst & 
monStringFirst & dayStringFirst & "_" & TowerString(i) & ".txt"
			If FSO.FileExists(LDrivePlotsFileString) Then
				FSO.DeleteFile LDrivePlotsFileString, True
			End If
		End If


		If gleFileOpen = 1 Then
			gleFile.close
			FSO.DeleteFile gleFileString, True
			gleFileOpen = 0
		End If

		If i <> 4 Then
			MsgBox("Are you ready to plot Tower " & i+2 & "?")
		End If

	Else	'*** If not data then create "NO DATA" text file for data and plots archive ***
		MsgBox("No Data for Tower " & i+1 & " - " & YMDFirst)

		If Not FSO.FolderExists(rootPath & plotsPath & YMDFirst) Then
			FSO.CreateFolder(rootPath & plotsPath & YMDFirst)
		End If
		plotsNoDataFileString = rootPath & plotsPath & YMDFirst & "\MSA_Plots_MST_" & yearStringFirst & monStringFirst & 
dayStringFirst & "_" & TowerString(i) & ".txt"
		set plotsNoDataFile = FSO.OpenTextFile(plotsNoDataFileString, ForWriting, True)
		plotsNoDataFile.WriteLine("No Data Files were Found.")
		plotsNoDataFile.close

		' *** Delete jpg file if one exists from both C and L Drive ***
		jpgFileArchiveString = rootPath & plotsPath & YMDFirst & "\MSA_Plots_MST_" & yearStringFirst & monStringFirst & 
dayStringFirst & "_" & TowerString(i) & ".jpg"
```

```
        LDrivePlotsFileString = LDriveRootPath & plotsPath & YMDFirst & "\MSA_Plots_MST_" & yearStringFirst & monStringFirst &
dayStringFirst & "_" & TowerString(i) & ".jpg"

        If FSO.FileExists(jpgFileArchiveString) Then
                FSO.DeleteFile jpgFileArchiveString, True
                'MsgBox("jpgFileArchiveString to delete=" & jpgFileArchiveString)
        End If
        If FSO.FileExists(LDrivePlotsFileString) Then
                FSO.DeleteFile LDrivePlotsFileString, True
        End If

        'If Not FSO.FolderExists(rootPath & archivePath & YMDFirst) Then
        '       FSO.CreateFolder(rootPath & archivePath & YMDFirst)
        'End If
        'archiveNoDataFileString = rootPath & archivePath & YMDFirst & "\MSA_MST_" & yearStringFirst & monStringFirst &
dayStringFirst & "_" & TowerString(i) & ".txt"
        'set archiveNoDataFile = FSO.OpenTextFile(archiveNoDataFileString, ForWriting, True)
        'archiveNoDataFile.WriteLine("No Data Files were Found.")
        'archiveNoDataFile.close

  End If    '*** If TowerDataFiles ***

 Next       '***For Loop Tower (i) ***

 LDrivePlotsDirString = LDriveRootPath & plotsPath
 FSO.CopyFolder rootPath & plotsPath & YMDFirst, LDrivePlotsDirString, True

 If k < 1 Then
        MsgBox("Are you ready to plot Tower 1 for " & YMD & "?")
        YMDFirst = YMD
        dayStringFirst = dayString
        monStringFirst = monString
        yearStringFirst = yearString
 End If


Next    '*** For Loop YMD and YMD+1 to plot data (k) ***
```

```
Function YMDFromJulian(ByRef yearString, ByRef monString, ByRef dayString, ByVal JulianDay)

yearRatio = CInt(yearString) / 4.0
yearRemainder = yearRatio - Int(yearRatio)


If (yearRemainder > 0.1) Then
'Non-leap year
  If (JulianDay >= 1) And (JulianDay <= 31) Then
    monString = "01"
    dayString = CStr(JulianDay)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 32) And (JulianDay <= 59) Then
    monString = "02"
    dayString = CStr(JulianDay - 31)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 60) And (JulianDay <= 90) Then
    monString = "03"
    dayString = CStr(JulianDay - 59)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 91) And (JulianDay <= 120) Then
    monString = "04"
    dayString = CStr(JulianDay - 90)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 121) And (JulianDay <= 151) Then
    monString = "05"
    dayString = CStr(JulianDay - 120)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 152) And (JulianDay <= 181) Then
    monString = "06"
    dayString = CStr(JulianDay - 151)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 182) And (JulianDay <= 212) Then
    monString = "07"
    dayString = CStr(JulianDay - 181)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 213) And (JulianDay <= 243) Then
    monString = "08"
    dayString = CStr(JulianDay - 212)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 244) And (JulianDay <= 273) Then
    monString = "09"
    dayString = CStr(JulianDay - 243)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 274) And (JulianDay <= 304) Then
    monString = "10"
    dayString = CStr(JulianDay - 273)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 305) And (JulianDay <= 334) Then
    monString = "11"
    dayString = CStr(JulianDay - 304)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 335) And (JulianDay <= 365) Then
    monString = "12"
    dayString = CStr(JulianDay - 334)
    dayString = string(2 - Len(dayString), "0") & dayString
  End If
Else
'Leap year
  If (JulianDay >= 1) And (JulianDay <= 31) Then
    monString = "01"
    dayString = CStr(JulianDay)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 32) And (JulianDay <= 60) Then
    monString = "02"
    dayString = CStr(JulianDay - 31)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 61) And (JulianDay <= 91) Then
    monString = "03"
    dayString = CStr(JulianDay - 60)
```

```
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 92) And (JulianDay <= 121) Then
   monString = "04"
   dayString = CStr(JulianDay - 91)
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 122) And (JulianDay <= 152) Then
   monString = "05"
   dayString = CStr(JulianDay - 121)
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 153) And (JulianDay <= 182) Then
   monString = "06"
   dayString = CStr(JulianDay - 152)
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 183) And (JulianDay <= 213) Then
   monString = "07"
   dayString = CStr(JulianDay - 182)
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 214) And (JulianDay <= 244) Then
   monString = "08"
   dayString = CStr(JulianDay - 213)
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 245) And (JulianDay <= 274) Then
   monString = "09"
   dayString = CStr(JulianDay - 244)
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 275) And (JulianDay <= 305) Then
   monString = "10"
   dayString = CStr(JulianDay - 274)
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 306) And (JulianDay <= 335) Then
   monString = "11"
   dayString = CStr(JulianDay - 305)
   dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 336) And (JulianDay <= 366) Then
   monString = "12"
   dayString = CStr(JulianDay - 335)
   dayString = string(2 - Len(dayString), "0") & dayString
  End If
End If

End Function
```

.

## Appendix F. Data Management Program – MSA_QC_MST.gle

This appendix appears in its original form, without editorial change.

The *MSA_QC_MST.gle* program creates a 24-h midnight to midnight Mountain Standard Time (MST) data displayof various Meteorological Sensor Array (MSA) data time series plots.

```
! PROGRAM:  MSA_QC_MST.gle
! AUTHOR:   O'Brien/Harrison
! Last Rev: 05-10-2014, sh
!
! PURPOSE:  This program creates a 24 hour midnight to midnight MST data display of various plots for MSA Data

!subroutine to find max value
sub dmaxy ds$
        local crmax = datayvalue(ds$,1)
        for i = 2 to ndata(ds$)
                crmax = max(crmax, datayvalue(ds$,i))
        next i
        !print "crmax: " crmax
        return crmax
end sub

size 63 60
set font ssb
set hei 0.7
set alabelscale 1.0
set atitlescale 1.0
set titlescale 1.0


!Find YYYYMMDD and XX:YY from file and then set up
! file date string and tower position for title info

dataFile$ = "C:\MSA\MSA_POST-PoC_Exercise\MSA_Data\MSA_24hr_GLE_Data.txt"
fopen dataFile$ f1 read
fgetline f1 line$
fclose f1
year$ = seg$(line$, 2, 5)
mon$ = seg$(line$, 6, 7)
day$ = seg$(line$, 8, 9)
position$ = seg$(line$, 23, 27)

monValue = val(mon$)
xcomp$ = seg$(position$, 2, 2)
ycomp$ = seg$(position$, 5, 5)
xcompValue = val(xcomp$)
ycompValue = val(ycomp$)

if (xcompValue = 1) then
        if (ycompValue = 1) then
                tower$ = "1 "
        else if (ycompValue = 2) then
                tower$ = "2 "
        else
                tower$ = "5 "
        end if
else if (xcompValue = 2) then
        tower$ = "3 "
else
        tower$ = "4 "
end if

if (monValue = 1) then
   mon$ = "Jan "
else if (monValue = 2) then
   mon$ = "Feb "
else if (monValue = 3) then
   mon$ = "Mar "
```

```
else if (monValue = 4) then
   mon$ = "Apr "
else if (monValue = 5) then
   mon$ = "May "
else if (monValue = 6) then
   mon$ = "Jun "
else if (monValue = 7) then
   mon$ = "Jul "
else if (monValue = 8) then
   mon$ = "Aug "
else if (monValue = 9) then
   mon$ = "Sep "
else if (monValue = 10) then
   mon$ = "Oct "
else if (monValue = 11) then
   mon$ = "Nov "
else if (monValue = 12) then
   mon$ = "Dec "
end if


!Output file date and tower position header
set hei 1.0
set color blue
amove 8 57.5
write "Data Date: " mon$ day$ ", " year$ " MST"
amove 42 57.5
write "Tower " tower$  "(" position$ ")"
set color black
set hei 0.7


!Wind speed plot at top of page

amove 1 30
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 20
YAxisMax30 = 30
YAxisMin = 0
YAxisMajorTick = 5
YAxisSubTick = 1

begin graph
  size 20 12
  nobox
  !title "Wind Speed (1 - Minute Average)"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  y2ticks length 0.3
  y2subticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick on
  ylabels on
  y2labels off
  ytitle "Wind Speed (m/s)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c7 d2=c2,c16
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
end graph

!Trying to resize yaxis depending on data
maxyd1 = dmaxy(d1)
```

```
maxyd2 = dmaxy(d2)
if (maxyd1 <= 10) and (maxyd2 <= 10) then
        !begin graph
                !yaxis min 0 max 10 dticks 2 dsubticks 1
        !end graph
end if

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 color brown
  separator
  text "10m" lstyle 1 color blue
end key
set hei 0.7
```

**!Output Graph Title wind speed**

```
amove 5 40.8
set color green
write "Wind Speed (m/s) \; Tower " tower$ "(" position$ ")"
set color black
```

**!Wind direction plot at top of page**

```
amove 1 44
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 360
YAxisMin = 0
YAxisMajorTick = 90
YAxisSubTick = 30

begin graph
  size 20 12
  nobox
  !title "Wind Direction (1 - Minute Average)"
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .2
  yticks length .2
  ysubticks length 0.3
  ylabels on
  y2labels off
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick grid dsubticks YAxisSubTick
  ytitle "Wind Direction (degrees)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c8 d2=c2,c17
  d1 marker circle msize 0.3 color brown
  d2 marker circle msize 0.3 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 color brown
  separator
  text "10m" lstyle 1 color blue
end key
set hei 0.7
```

**!Output Graph Title wind direction**

```
amove 3.5 54.8
set color green
write "Wind Direction: (degrees) \; Tower " tower$ "(" position$ ")"
set color black
```

**!U plot at bottom of page**

```
amove 1 16
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 15
YAxisMax30 = 30
YAxisMin = -10
YAxisMajorTick = 5
YAxisSubTick = 1

begin graph
  size 20 12
  nobox
  !title "2m U Plot"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  y2ticks length 0.3
  y2subticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick on
  ylabels on
  y2labels off
  ytitle "U (m/s)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c9 d2=c2,c18
  let d3 = d1*0.0
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
  d3 lstyle 1 lwidth .1 color deeppink
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 color brown
  separator
  text "10m" lstyle 1 color blue
end key
set hei 0.7
```

**!Output Graph Title of U Component**

```
amove 5 26.8
set color green
write "U Component (m/s) \; Tower " tower$ "(" position$ ")"
set color black
```

**!V plot at bottom of page**

```
amove 22 16
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 10
YAxisMax30 = 30
YAxisMin = -10
YAxisMajorTick = 5
```

```
YAxisSubTick = 1

begin graph
  size 20 12
  nobox
  !title "2m V Plot"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  y2ticks length 0.3
  y2subticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick on
  ylabels on
  y2labels off
  ytitle "V (m/s)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c10 d2=c2,c19
  let d3 = d1*0.0
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
  d3 lstyle 1 lwidth .1 color deeppink
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 color brown
  separator
  text "10m" lstyle 1 color blue
end key
set hei 0.7
```

**!Output Graph Title for V Component**

```
amove 25.5 26.8
set color green
write "V Component (m/s) \; Tower " tower$ "(" position$ ")"
set color black
```

**!W plot at bottom of page**

```
amove 43 16
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 1
YAxisMax30 = 30
YAxisMin = -1
YAxisMajorTick = .5
YAxisSubTick = .1

begin graph
  size 20 12
  nobox
  !title "2m W Plot"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  y2ticks length 0.3
  y2subticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick on
  ylabels on
  y2labels off
  ytitle "W (m/s)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c11 d2=c2,c20
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 color brown
  separator
  text "10m" lstyle 1 color blue
end key
set hei 0.7

!Output Graph Title for W Component
amove 46.5 26.8
set color green
write "W Component (m/s) \; Tower " tower$ "(" position$ ")"
set color black


!Errors plot at bottom of page

amove 1 2
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 1200
YAxisMax30 = 30
YAxisMin = 0
YAxisMajorTick = 300
YAxisSubTick = 100

begin graph
  size 20 12
  nobox
  !title "Errors"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.3
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  ylabels on
  ytitle "# of Errors"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c14 d2=c2,c23
  d1 lstyle 1 lwidth .05 color brown
  d2 lstyle 1 lwidth .05 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2m" lstyle 1 color brown
  separator
  text "10m" lstyle 1 color blue
end key
set hei 0.7

!Output Graph Title for Errors
```

```
amove 4 12.8
set color green
write "Number of Sonic Errors \; Tower " tower$ "(" position$ ")"
set color black
```

**!Dew Point plot at bottom of page**

```
amove 22 2
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 50
YAxisMin = -20
!YAxisMajorTick = 10
YAxisMajorTick = 5
YAxisSubTick = 1

begin graph
  size 20 12
  nobox
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length .2
  !yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick grid dsubticks YAxisSubTick
  yaxis dticks YAxisMajorTick grid dsubticks YAxisSubTick
  ylabels on
  y2labels off
  ytitle "Temperature ( ^{o}C)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c6 d3=c2,c4
  let d4 = (d3*0.06112*EXP((17.67*d1)/(d1+243.5)))
  let d5 = ((243.5*LOG(d4/6.112))/(17.67-LOG(d4/6.112)))
  let d6 = d1*0.0
  d5 lstyle 1 lwidth .04 color green
  d6 lstyle 1 lwidth .1 color deeppink
end graph
```

**!Output Graph Title for DewPoint**

```
amove 27 12.8
set color green
write "DewPoint (^{o}C) \; Tower " tower$ "(" position$ ")"
set color black
```

**!Battery Voltage and Panel Temp plot at bottom of page**

```
amove 43 2
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 15
YAxisMax30 = 30
YAxisMin = 10
YAxisMajorTick = 1
YAxisSubTick = 0

begin graph
  size 20 12
  nobox
  !title "Plot"
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  !ysubticks length 0.3
  y2ticks length 0.5
  y2subticks length 0.5
  !yplaces 10 11 12 13 14 15 16
  !ynames "10" "11" "12" "13" "14" "15" "16"
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis min YAxisMin dticks YAxisMajorTick dsubticks YAxisSubTick
  y2axis min 0 max 60 dticks 12 dsubticks 6
  ylabels on
  y2labels on
  ytitle "Battery Voltage (V)"
  xtitle "MST (decimal hours)" dist 0.5
  y2title "Panel Temp ( ^{o}C)"
  data dataFile$ d1=c2,c24 d2=c2,c25
  let d2 = (d2/12)+10
  d1 lstyle 1 lwidth .05 color brown
  d2 lstyle 1 lwidth .05 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "Battery" lstyle 1 color brown
  separator
  text "Panel-T" lstyle 1 color blue
end key
set hei 0.7
```

**!Output Graph Title for Battery Voltage and Panel Temp**

```
amove 45 12.8
set color green
write "Battery(V) \; Panel-T(^{o}C) \; Tower " tower$ "(" position$ ")"
set color black
```

**!Temperature gradient plot at top of page**

```
amove 22 30
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = .9
YAxisMin = -0.4
YAxisMajorTick = 0.1
YAxisSubTick = 0.05

begin graph
  size 20 12
  nobox
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length 0.2
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  !yaxis dticks YAxisMajorTick dsubticks YAxisSubTick
  ylabels on
  y2labels off
  ytitle "Temperature Gradient ( ^{o}C/m)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c6 d2=c2,c15
  let d3 = (d2-d1)/8
  let d4 = d1*0.0
  d3 lstyle 1 color purple
  d4 lstyle 1 lwidth .1 color deeppink
end graph
```

**!Output graph title for temperature gradient**

```
amove 25 40.8
```

```
set color green
write "Temp Gradient (^{o}C/m) \; Tower " tower$ "(" position$ ")"
set color black



!Temperature plot at top of page

amove 22 44
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 45
YAxisMin = 10
YAxisMajorTick = 5
YAxisSubTick = 0

begin graph
  size 20 12
  nobox
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .3
  ysubticks length .2
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick grid dsubticks YAxisSubTick
  !yaxis dticks YAxisMajorTick grid dsubticks YAxisSubTick
  ylabels on
  y2labels off
  ytitle "Temperature ( ^{o}C)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c6 d2=c2,c15
  d1 lstyle 1 color brown
  d2 lstyle 1 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "2 m" lstyle 1 color brown
  separator
  text "10 m" lstyle 1 color blue
  end key
set hei 0.7

!Output Graph Title for Temperature
amove 26 54.8
set color green
write "Temperature (^{o}C) \; Tower " tower$ "(" position$ ")"
set color black


!Solar irradiance plot at top of page

amove 43 30
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 1400
YAxisMin = 0
YAxisMajorTick = 200
YAxisSubTick = 100

begin graph
  size 20 12
  nobox
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .2
  yticks length .2
  ysubticks length 0.3
  ylabels on
  y2labels off
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  ytitle "Solar Irradiance (W/m^2)"
  xtitle "MST (decimal hours)" dist 0.5
  data dataFile$ d1=c2,c5
  d1 lstyle 1 lwidth .04 color brown
end graph

!Output graph title for solar irradiance
amove 46 40.8
set color green
write "Solar Irradiance (W/m^2) \; Tower " tower$ "(" position$ ")"
set color black


!Relative humidity and station air pressure plot at top of page

amove 43 44
XAxisMax = 24
XAxisMin = 0
XAxisMajorTick = 6.0
XAxisSubTick = 1.0
YAxisMax = 100
YAxisMin = 0
YAxisMajorTick = 10
YAxisSubTick = 5.0

begin graph
  size 20 12
  nobox
  yaxis grid
  xaxis min XAxisMin max XAxisMax dticks XAxisMajorTick dsubticks XAxisSubTick
  xticks length .3
  yticks length .5
  y2ticks length .5
  yaxis min YAxisMin max YAxisMax dticks YAxisMajorTick dsubticks YAxisSubTick
  ytitle "Relative Humidity (percent)"
  xtitle "MST (decimal hours)" dist 0.5
  y2axis min 850 max 880 dticks 10 dsubticks 0
  y2title "Station Air Pressure (mb)"
  y2labels on
  data dataFile$ d1=c2,c4 d2=c2,c3
  let d2 = (d2-850)*3.3333
  d1 lstyle 1 lwidth .04 color brown
  d2 lstyle 1 lwidth .04 color blue
end graph

begin key
  hei 0.5
  position bc
  offset 0.0 -2.3
  text "Humidity" lstyle 1 color brown
  separator
  text "Pressure" lstyle 1 color blue
end key
set hei 0.7

!Output graph title for relative humidity
amove 45 54.8
set color green
write "RH (%) \; Pressure (mb) \; Tower " tower$ "(" position$ ")"
set color black
```

INTENTIONALLY LEFT BLANK.

## Appendix G. Data Management Program – Met_ASCII_UTC.vbs

This appendix appears in its original form, without editorial change.

The *Met_ASCII_UTC.vbs* program reads Meteorological Sensor Array (MSA) merged hourly files, and formats them into MET American Standard Code for Information Interchange (ASCII) to be read into the MET V&V software.

```
' PROGRAM:  Met_ASCII_UTC.vbs
' AUTHOR:   Sandra Harrison
' Last Rev: 140417, gv/sh
'
' PURPOSE:  This program reads MSA merged hourly files and formats them into Met ASCII to be read into the VVA Software.
' There will be 25 hourly output files (6am to 6am) and each file will include calculated data for each tower.


Dim metAsciiFileString
Dim sourceFileString
Dim FSO
Dim sourceFile
Dim metAsciiFile
Dim currentLineNumber

Const ForReading = 1, ForWriting = 2, ForAppending = 8

set FSO = CreateObject("Scripting.FileSystemObject")
set WshShell = CreateObject("WScript.Shell")

rootPath = "L:\"
dataSourcePath = "MSA_POST-PoC_Exercise\MSA_Data\"
archivePath = "MSA_SH\MET_ASCII\ArchiveData\"
applicationsPath = "MSA_SH\MET_ASCII\"

Set args = WScript.Arguments
YMD = args.Item(0)

'YMD = InputBox("Enter YEAR/MONTH/DAY of Data Files in Format (YYMMDD):", "UTC")
If YMD="" Then
          WScript.Quit
End If

yearString = Left(YMD, 2)
monString = Mid(YMD, 3, 2)
dayString = Mid(YMD, 5, 2)
YMDOrig = YMD

Dim towerString
towerString = Array("0101", "0102", "0202", "0302", "0103")
utcString = "05"

For k = 0 to 24          '***Hours***

metAsciiFileOpen = 0

For i = 0 To 4           '***Towers***

  Continue = True
  sourceFileString = rootPath & dataSourcePath & YMD & "merged\20" & yearString & monString & dayString & "_" & utcString & "00" &
"_" & towerString(i) & "_merged.txt"

  If FSO.FileExists(sourceFileString) Then
          set sourceFile = FSO.OpenTextFile(sourceFileString, ForReading, True)        '***Open hourly file of tower for reading***
          'MsgBox("Data File:  " & sourceFileString)
  Else
          x = MsgBox("File Not Found: " & sourceFileString, 0, "WARNING")
          'WScript.Quit
          Continue = False
  End If
```

```
If Continue Then                    '***If Data Exists Then Read Data***
  currentLineNumber = 0
  Total_Pressure = 0
  Total_RH = 0
  Total_Temp2m = 0
  Total_WS = 0
  Total_U = 0
  Total_V = 0
  Press15mincnt = 0
  RH15mincnt = 0
  Temp15mincnt = 0
  WS15mincnt = 0
  U15mincnt = 0
  V15mincnt = 0

  UTCCol = 1
  PressureCol = 3
  RHCol = 4
  Temp2mCol = 6
  WSCol = 16
  UCol = 18
  VCol = 19

  Do Until sourceFile.AtEndOfStream
        CurCol = 0
        LogicalCol = 0
        currentLineNumber = currentLineNumber + 1

        currentLine = sourceFile.ReadLine
        currentLineArray = Split(currentLine, " ")

        If currentLineNumber = 1 Then
                julianDayString = currentLineArray(1)
                Latitude = currentLineArray(5)
                'Wscript.Echo "Latitude: " & currentLineArray(5)
                Longitude = currentLineArray(6)
                'Wscript.Echo "Longitude: " & currentLineArray(6)
                Elevation = CDbl(currentLineArray(7))
                'Wscript.Echo "Elevation: " & currentLineArray(7)
        Else
                For j = 0 to 20                 '***Read each line of file accounting for extra spaces***
                        Do While currentLineArray(CurCol) = "" OR currentLineArray(CurCol) = Space(1)
                                CurCol = CurCol + 1
                        Loop

                        LogicalCol = LogicalCol + 1
                        'Wscript.Echo "LogicalCol:" & LogicalCol & "  CurCol:" & CurCol & "  J:" & j

                        If UTCCol = LogicalCol  Then
                                UTC = CDbl(currentLineArray(CurCol))
                                'Wscript.Echo "UTC:" & currentLineArray(CurCol)
                        End If
                        If PressureCol = LogicalCol  Then
                                Pressure = CDbl(currentLineArray(CurCol))
                                If CDbl(Pressure) <= -99 Then
                                        Pressure = -9999
                                End If
                                'Wscript.Echo "Pressure: " & currentLineArray(CurCol)
                        End If
                        If RHCol = LogicalCol  Then
                                RH = currentLineArray(CurCol)
                                If CDbl(RH) <= -99 Then
                                        RH = -9999
                                End If
                                'Wscript.Echo "RH: " & currentLineArray(CurCol)
                        End If
                        If Temp2mCol = LogicalCol  Then
                                Temp2m = currentLineArray(CurCol)
                                If CDbl(Temp2m) <= -99 Then
                        Temp2m = -9999
                    End If
                    'Wscript.Echo "Temp2m: " & currentLineArray(CurCol)
                End If
                If WSCol = LogicalCol  Then
                    WindSpeed = currentLineArray(CurCol)
                    If CDbl(windSpeed) <= -99 Then
                        windSpeed = -9999
                    End If
                    'Wscript.Echo "WindSpeed: " & currentLineArray(CurCol)
                End If
                If UCol = LogicalCol  Then
                    UComp = currentLineArray(CurCol)
                    If CDbl(UComp) <= -99 Then
                        UComp = -9999
                    End If
                    'Wscript.Echo "UComp: " & currentLineArray(CurCol)
                End If
                If VCol = LogicalCol  Then
                    VComp = currentLineArray(CurCol)
                    If CDbl(VComp) <= -99 Then
                        VComp = -9999
                    End If
                    'Wscript.Echo "VComp: " & currentLineArray(CurCol)
                End If
                CurCol = CurCol + 1
            Next

            If currentLineNumber = 2 Then
                UTC_last15min = UTC + .75
            End If


            If CDbl(UTC) >= CDbl(UTC_last15min) Then                    '***Sum Totals***
                If Pressure <> -9999  Then
                    Total_Pressure = Total_Pressure + CDbl(Pressure)
                    Press15mincnt = Press15mincnt + 1
                    'Wscript.Echo "Pressure: " & Pressure
                    'Wscript.Echo "Total_Pressure: " & Total_Pressure
                    'Wscript.Echo "fifteenMinCount: " & fifteenMinCount
                End If

                If RH <> -9999  Then
                    Total_RH = Total_RH + CDbl(RH)
                    RH15mincnt = RH15mincnt + 1
                    'Wscript.Echo "RH: " & RH
                    'Wscript.Echo "Total_RH " & Total_RH
                End If

                If Temp2m <> -9999  Then
                    Total_Temp2m = Total_Temp2m + CDbl(Temp2m)
                    Temp15mincnt = Temp15mincnt + 1
                End If

                If WindSpeed <> -9999  Then
                    Total_WS = Total_WS + CDbl(WindSpeed)
                    WS15mincnt = WS15mincnt + 1
                End If

                If UComp <> -9999  Then
                    Total_U = Total_U + CDbl(UComp)
                    U15mincnt = U15mincnt + 1
                End If

                If VComp <> -9999  Then
                    Total_V = Total_V + CDbl(VComp)
                    V15mincnt = V15mincnt + 1
                End If
            End If
        End If
```

```
Loop


If Total_Pressure = 0 OR Press15mincnt < 14 Then                    '***Calculate Averages for last 15 mins of hour***
        Total_Pressure = -9999
Else
        Total_Pressure = CDbl(Total_Pressure) / Press15mincnt
        GetAltimeter AltPress, Total_Pressure, Elevation
        AltPressPa = AltPress * 100                                 '***Conversion mb to pa***
End If

If Total_RH <> 0  Then
        If RH15mincnt < 14 Then
                Total_RH = 0
        Else
                Total_RH = Total_RH / RH15mincnt
        End If
End If

If Total_Temp2m = 0 OR Temp15mincnt < 14  Then
        Total_Temp2mK = 0
Else
        Total_Temp2m = Total_Temp2m / Temp15mincnt
        Total_Temp2mK = Total_Temp2m + 273.15                               '***Conversion Celsius to Kelvin***
End If

If Total_Temp2mK <> 0 AND Total_RH <> 0  Then
        GetDewPoint DewPoint, Total_Temp2mK, Total_RH
End If

If Total_WS <> 0  Then
        If WS15mincnt < 14 Then
                Total_WS = 0
        Else
                Total_WS = Total_WS / WS15mincnt
        End If
End If

If Total_U <> 0 Then
        If U15mincnt < 14 Then
                Total_U = 0
        Else
                Total_U = Total_U / U15mincnt
        End If
End If

If Total_V <> 0 Then
        If V15mincnt < 14 Then
                Total_V = 0
        Else
                Total_V = Total_V / V15mincnt
        End If
End If



metAsciiUtcString = CStr(CInt(utcString)+1)                                 '***save file to top of hour value***
metAsciiUtcString = string(2 - Len(metAsciiUtcString), "0") & metAsciiUtcString
If metAsciiUtcString = "24" Then
        metAsciiUtcString = "00"
        If i = 0 Then
                julianDayString = CStr(CInt(julianDayString)+1)
                return = YMDFromJulian(yearString, monString, dayString, Int(julianDayString))
                YMD = yearString & monString & dayString
                'MsgBox("Next day: " & yearString & monString & dayString)
        End If
End If

If Not FSO.FolderExists(rootPath & archivePath & YMDOrig) Then
        FSO.CreateFolder(rootPath & archivePath & YMDOrig)
```

```
  End If

              '***Open MetAsciiFile to print calculated data for tower # for hour #***
  metAsciiFileString = rootPath & archivePath & YMDOrig & "\hr" & CStr(k) & "_" & metAsciiUtcString & "Z.txt"
  If metAsciiFileOpen = 0  Then
              set metAsciiFile = FSO.OpenTextFile(metAsciiFileString, ForWriting, True)
              metAsciiFileOpen = 1
              'MsgBox("MetAscii File:  " & metAsciiFileString)
  Else
              metAsciiFile.close
              set metAsciiFile = FSO.OpenTextFile(metAsciiFileString, ForAppending, True)
              'MsgBox("MetAscii File:  " & metAsciiFileString)
  End If


              '***Write Data to File***
  If Total_Temp2mK <> 0  Then
              metAsciiFile.WriteLine("ADPSFC " & "MSA" & towerString(i) & " 20" & yearString & monString & dayString & "_" &
metAsciiUtcString & "0000 " & Latitude & " " & Longitude & " " & FormatNumber(Elevation, 2, , , vbFalse) & " 11 " &
FormatNumber(Total_Pressure, 4, , , vbFalse) & " " & FormatNumber(Elevation, 2, , , vbFalse) & " NA " & FormatNumber(Total_Temp2mK, 6,
, , vbFalse))
  End If

  If Total_Temp2mK <> 0 AND Total_RH <> 0  Then
              metAsciiFile.WriteLine("ADPSFC " & "MSA" & towerString(i) & " 20" & yearString & monString & dayString & "_" &
metAsciiUtcString & "0000 " & Latitude & " " & Longitude & " " & FormatNumber(Elevation, 2, , , vbFalse) & " 17 " &
FormatNumber(Total_Pressure, 4, , , vbFalse) & " " & FormatNumber(Elevation, 2, , , vbFalse) & " NA " & FormatNumber(DewPoint, 6, , ,
vbFalse))
  End If

  If Total_Pressure <> -9999  Then
              metAsciiFile.WriteLine("ADPSFC " & "MSA" & towerString(i) & " 20" & yearString & monString & dayString & "_" &
metAsciiUtcString & "0000 " & Latitude & " " & Longitude & " " & FormatNumber(Elevation, 2, , , vbFalse) & " 2 " &
FormatNumber(Total_Pressure, 4, , , vbFalse) & " " & FormatNumber(Elevation, 2, , , vbFalse) & " NA " & FormatNumber(AltPressPa, 6, , ,
vbFalse))
  End If

  If Total_WS <> 0  Then
              metAsciiFile.WriteLine("ADPSFC " & "MSA" & towerString(i) & " 20" & yearString & monString & dayString & "_" &
metAsciiUtcString & "0000 " & Latitude & " " & Longitude & " " & FormatNumber(Elevation, 2, , , vbFalse) & " 32 " &
FormatNumber(Total_Pressure, 4, , , vbFalse) & " " & FormatNumber(Elevation, 2, , , vbFalse) & " NA " & FormatNumber(Total_WS, 6, , ,
vbFalse))
  End If

  If Total_U <> 0  Then
              metAsciiFile.WriteLine("ADPSFC " & "MSA" & towerString(i) & " 20" & yearString & monString & dayString & "_" &
metAsciiUtcString & "0000 " & Latitude & " " & Longitude & " " & FormatNumber(Elevation, 2, , , vbFalse) & " 33 " &
FormatNumber(Total_Pressure, 4, , , vbFalse) & " " & FormatNumber(Elevation, 2, , , vbFalse) & " NA " & FormatNumber(Total_U, 6, , ,
vbFalse))
  End If

  If Total_V <> 0  Then
              metAsciiFile.WriteLine("ADPSFC " & "MSA" & towerString(i) & " 20" & yearString & monString & dayString & "_" &
metAsciiUtcString & "0000 " & Latitude & " " & Longitude & " " & FormatNumber(Elevation, 2, , , vbFalse) & " 34 " &
FormatNumber(Total_Pressure, 4, , , vbFalse) & " " & FormatNumber(Elevation, 2, , , vbFalse) & " NA " & FormatNumber(Total_V, 6, , ,
vbFalse))
  End If

  If Total_RH <> 0  Then
              metAsciiFile.WriteLine("ADPSFC " & "MSA" & towerString(i) & " 20" & yearString & monString & dayString & "_" &
metAsciiUtcString & "0000 " & Latitude & " " & Longitude & " " & FormatNumber(Elevation, 2, , , vbFalse) & " 52 " &
FormatNumber(Total_Pressure, 4, , , vbFalse) & " " & FormatNumber(Elevation, 2, , , vbFalse) & " NA " & FormatNumber(Total_RH, 6, , ,
vbFalse))
  End If

  metAsciiFile.WriteLine("ADPSFC " & "MSA" & towerString(i) & " 20" & yearString & monString & dayString & "_" & metAsciiUtcString
& "0000 " & Latitude & " " & Longitude & " " & FormatNumber(Elevation, 2, , , vbFalse) & " 7 " & FormatNumber(Total_Pressure, 4, , ,
vbFalse) & " " & FormatNumber(Elevation, 2, , , vbFalse) & " NA " & FormatNumber(Elevation, 2, , , vbFalse))

  sourceFile.close
```

```
End If

Next

utcString = CStr(CInt(utcString)+1)                    '***set name of next source file to read from***
utcString = string(2 - Len(utcString), "0") & utcString
If utcString = "24" Then
        utcString = "00"
End If

metAsciiFile.close

Next


Function GetDewPoint(ByRef DewPoint, ByVal TempK, ByVal RH)

If TempK > 150 Then
        Temp = TempK - 273.15
Else
        Temp = TempK
End If
X = 1.0 - 0.01 * RH

DPD = (14.55 + 0.114 * Temp) * X + ((2.5 + 0.007 * Temp) * X) ^ 3 + (15.9 + 0.117 * Temp) * X ^ 14
DWPTC = Temp - DPD
DewPoint = DWPTC + 273.15

End Function


Function GetAltimeter(ByRef PressAlt, ByVal Press, ByVal Elev)

EXPONT = 0.190284
EXPINV = 1.0/EXPONT
LAPSE = 0.0065
TSTD = 288.15
PSTD = 1013.25
CONSTANT = PSTD ^ EXPONT * LAPSE / TSTD


PressAlt = ((Press - 0.3) ^ EXPONT + CONSTANT * Elev) ^ EXPINV

End Function



Function YMDFromJulian(ByRef yearString, ByRef monString, ByRef dayString, ByVal JulianDay)

yearRatio = CInt(yearString) / 4.0
yearRemainder = yearRatio - Int(yearRatio)


If (yearRemainder > 0.1) Then
'Non-leap year
  If (JulianDay >= 1) And (JulianDay <= 31) Then
    monString = "01"
    dayString = CStr(JulianDay)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 32) And (JulianDay <= 59) Then
    monString = "02"
    dayString = CStr(JulianDay - 31)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 60) And (JulianDay <= 90) Then
    monString = "03"
    dayString = CStr(JulianDay - 59)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 91) And (JulianDay <= 120) Then
    monString = "04"
    dayString = CStr(JulianDay - 90)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 121) And (JulianDay <= 151) Then
    monString = "05"
    dayString = CStr(JulianDay - 120)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 152) And (JulianDay <= 181) Then
    monString = "06"
    dayString = CStr(JulianDay - 151)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 182) And (JulianDay <= 212) Then
    monString = "07"
    dayString = CStr(JulianDay - 181)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 213) And (JulianDay <= 243) Then
    monString = "08"
    dayString = CStr(JulianDay - 212)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 244) And (JulianDay <= 273) Then
    monString = "09"
    dayString = CStr(JulianDay - 243)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 274) And (JulianDay <= 304) Then
    monString = "10"
    dayString = CStr(JulianDay - 273)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 305) And (JulianDay <= 334) Then
    monString = "11"
    dayString = CStr(JulianDay - 304)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 335) And (JulianDay <= 365) Then
    monString = "12"
    dayString = CStr(JulianDay - 334)
    dayString = string(2 - Len(dayString), "0") & dayString
  End If
Else
'Leap year
  If (JulianDay >= 1) And (JulianDay <= 31) Then
    monString = "01"
    dayString = CStr(JulianDay)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 32) And (JulianDay <= 60) Then
    monString = "02"
    dayString = CStr(JulianDay - 31)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 61) And (JulianDay <= 91) Then
    monString = "03"
    dayString = CStr(JulianDay - 60)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 92) And (JulianDay <= 121) Then
    monString = "04"
    dayString = CStr(JulianDay - 91)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 122) And (JulianDay <= 152) Then
    monString = "05"
    dayString = CStr(JulianDay - 121)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 153) And (JulianDay <= 182) Then
    monString = "06"
    dayString = CStr(JulianDay - 152)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 183) And (JulianDay <= 213) Then
    monString = "07"
    dayString = CStr(JulianDay - 182)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 214) And (JulianDay <= 244) Then
    monString = "08"
    dayString = CStr(JulianDay - 213)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 245) And (JulianDay <= 274) Then
```

```
    monString = "09"
    dayString = CStr(JulianDay - 244)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 275) And (JulianDay <= 305) Then
    monString = "10"
    dayString = CStr(JulianDay - 274)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 306) And (JulianDay <= 335) Then
    monString = "11"
    dayString = CStr(JulianDay - 305)
    dayString = string(2 - Len(dayString), "0") & dayString
  ElseIf (JulianDay >= 336) And (JulianDay <= 366) Then
    monString = "12"
    dayString = CStr(JulianDay - 335)
    dayString = string(2 - Len(dayString), "0") & dayString
  End If
End If

End Function
```

INTENTIONALLY LEFT BLANK..

# Appendix H. Data Management Program – MET_ASCII_UTC.wbt

This appendix appears in its original form, without editorial change.

The *MET_ASCII_UTC.wbt* program prompts the user for the date and calls *MET-ASCII_UTC.vbs*, using a WinBatch graphical user interface (GUI).

```
; PROGRAM:  MET_ASCII_UTC.wbt
; AUTHOR:   Sandra Harrison
; Last Rev: 140606, sh
; PURPOSE:  This program prompts the user for the Date and calls
; MET_ASCII_UTC.vbs using a WINBATCH GUI

YMD = AskLine("UTC", "Enter YEAR/MONTH/DAY of Data Files in Format
(YYMMDD)", "", 0)

Run ("L:\MSA_SH\MET_ASCII\MET_ASCII_UTC.vbs", YMD)

ErrorMode(@OFF)
WinActivate ("~WARNING")
If !WinExist ("~WARNING")
     Message(YMD, "MSA data files reformatted to MET ASCII.")
Else
     While (WinExist ("~WARNING"))
          TimeDelay(5)
     EndWhile
     Message(YMD, "MET ASCII files may be incomplete due to
WARNINGS.")
ErrorMode(@CANCEL)
```

## List of Symbols, Abbreviations, and Acronyms

| | |
|---|---|
| 24/7 | 24 h/day–7 days/week |
| 3DWF | Three-Dimensional Wind Field |
| AC | alternating current |
| AGL | above ground level |
| Ah | ampere-hour |
| API | Application Program Interface |
| ARL | US Army Research Laboratory |
| ASCII | American Standard Code for Information Interchange |
| BASC | Board on Atmospheric Sciences and Climate |
| DAS | data acquisition system |
| DC | direct current |
| DTC | Developmental Testbed Center |
| GIS | Geographic Information System |
| GLE | Graphic Layout Engine |
| GPS | global positioning system |
| GRIB | Gridded Binary |
| GUI | graphical user interface |
| LAPS | Local Analysis and Prediction System |
| MADIS | Meteorological Assimilation Data Ingest System |
| MET | Model Evaluation Tools |
| MPPT | maximum power point tracking |
| MSA | Meteorological Sensor Array |
| MST | Mountain Standard Time |
| NCEP | National Centers for Environmental Prediction |

| | |
|---|---|
| NOAA | National Oceanic and Atmospheric Administration |
| NRC | National Research Council |
| NTP | Network Time Protocol |
| PV | Photovoltaic |
| QA | Quality Assurance |
| R&D | Research and Development |
| RTMA | Real-Time Mesoscale Analysis |
| USB | Universal Serial Bus |
| V&V | validation and verification |
| whr | watt-hours |
| WRE-N | Weather Running Estimate-Nowcast |
| WRF | Weather Research and Forecasting |
| WSMR | White Sands Missile Range |

| | |
|---|---|
| 1<br>(PDF) | DEFENSE TECHNICAL<br>INFORMATION CTR<br>DTIC OCA |
| 2<br>(PDF) | DIRECTOR<br>US ARMY RSRCH LAB<br>RDRL CIO LL<br>IMAL HRA MAIL & RECORDS<br>MGMT |
| 1<br>(PDF) | GOVT PRINTG OFC<br>A MALHOTRA |
| 6<br>(CD) | US ARMY RSRCH LAB<br>ATTN RDRL CIE M<br>BLDG 1622<br>WSMR NM 88002<br>R DUMAIS<br>T FOLEY<br>T JAMESON<br>D KNAPP<br>J RABY<br>J SMITH |
| 4<br>(CD) | US ARMY RSRCH LAB<br>ATTN RDRL CIE D<br>BLDG 1622<br>WSMR NM 88002<br>R BRICE<br>S D'ARCY<br>S HARRISON<br>J SWANSON |
| 10<br>(5 CD,<br>5 HC) | US ARMY RSRCH LAB<br>G VAUCHER<br>ATTN RDRL CIE D<br>BLDG 1622<br>WSMR NM 88002 |
| 1<br>(CD) | US ARMY RSRCH LAB<br>P CLARK<br>ATTN RDRL CIE<br>2800 POWDER MILL RD<br>ADELPHI MD 20783-1138 |
| 1<br>(CD) | US ARMY RSRCH LAB<br>S O BRIEN<br>ATTN RDRL CIE D<br>BLDG 1622<br>WSMR NM 88002 |
| 1<br>(CD) | DR J MCLAY<br>NAVAL RESEARCH LABORATORY<br>7 GRACE HOPPER AVE STOP 2<br>MONTEREY CA 93943 |
| 1<br>(CD) | R CRAIG DAF CIVILIAN<br>HQ AFWA 2WXG 16WS/WXN<br>101 NELSON DRIVE<br>OFFUTT AFB NE 68113-1023 |
| 1<br>(CD) | ARMY JOINT SUPPORT TEAM<br>SFAE IEW&S DCGS A<br>ATTN G BARNES<br>238 HARSTON ST BLDG 90060<br>HURLBURT FIELD FL 32544 |
| 1<br>(CD) | J STALEY<br>ARMY WEATHER PROPONENT<br>OFFICE<br>INTEGRATION<br>SYNCHRONIZATION AND<br>ANALYSIS (CDID)<br>US ARMY INTELLIGENCE<br>CENTER OF EXCELLENCE<br>550 CIBEQUE ST BLDG 61730<br>FT HUACHUCA AZ 85613 |

INTENTIONALLY LEFT BLANK.